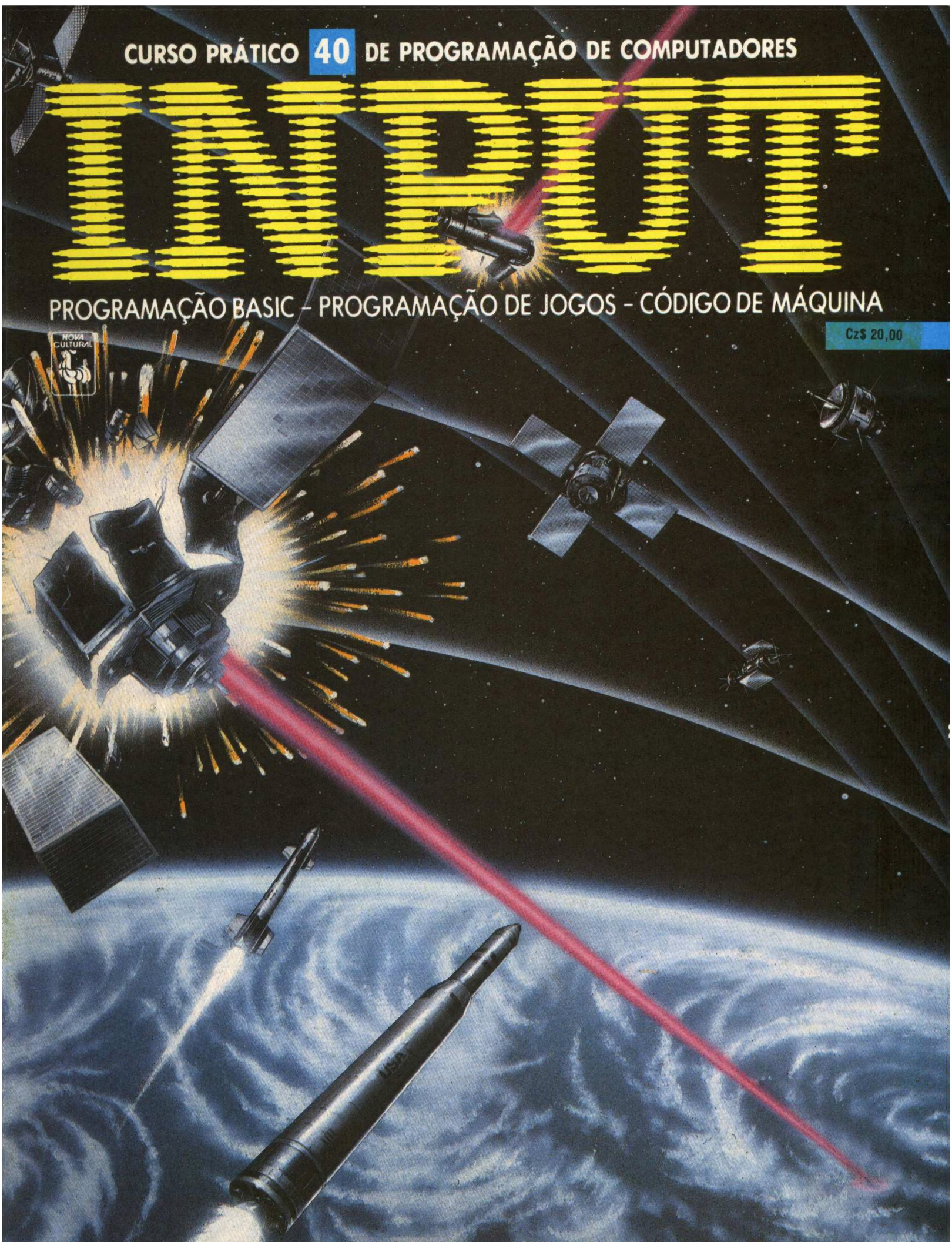
CURSO PRÁTICO 40 DE PROGRAMAÇÃO DE COMPUTADORES

INPUT

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

NOVA CULTURAL

Cz$ 20,00



INPUT

Vol. 3 N.º 40

NESTE NÚMERO

PROGRAMAÇÃO BASIC

SIMULAÇÕES ESPACIAIS

Trajetória de objetos que se movem em campos de baixa gravidade. Simulação de corpos em órbita. Acerte o alvo: cálculo da velocidade e do ângulo de lançamento. Circundando a Terra. Manobras no espaço. Movimento planetário 781

CÓDIGO DE MÁQUINA

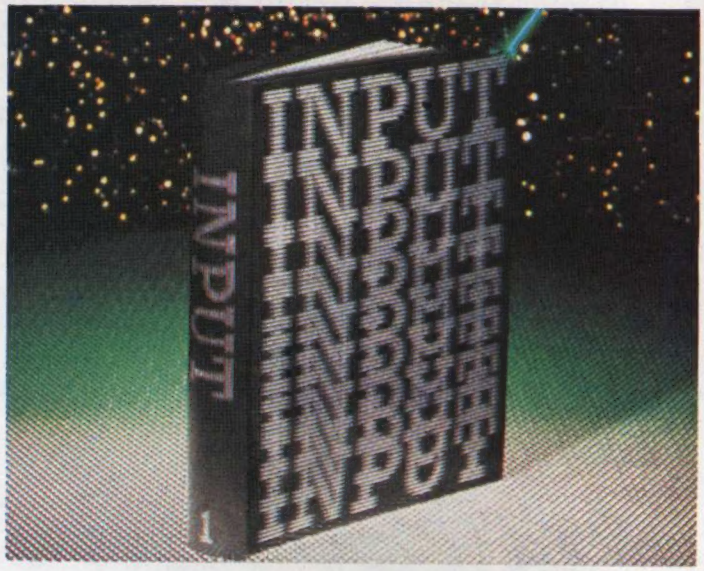
AVALANCHE: EFEITOS SONOROS

A escolha da música. Rotina de execução: como funciona. Contagem das notas. Cálculo das freqüências. Armazenagem da música. Como modificar a melodia. Como sincronizar o som e a ação do jogo. O uso do PSG, o microprocessador especial do MSX. Acordes. Controle da duração do acorde. Tabela de notas..... 788

PROGRAMAÇÃO DE JOGOS

O JOGO DO OTELO (2)

Movimentação do jogador. Verificação das posições. Rotina de movimentação do computador. Final da rotina de jogo. Anúncio do vencedor. Opção para nova partida..... 796



PLANO DA OBRA

"INPUT" é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

COMPLETE SUA COLEÇÃO

Exemplares atrasados, até seis meses após o encerramento da coleção, poderão ser comprados, a preços atualizados, da seguinte forma: 1. PESSOALMENTE — Por meio de seu jornaleiro ou dirigindo-se ao distribuidor local, cujo endereço poderá ser facilmente conseguido junto a qualquer jornaleiro de sua cidade. Em **São Paulo**, os endereços são: rua Brigadeiro Tobias, 773, Centro; av. Industrial, 117, Santo André; e no **Rio de Janeiro**: rua da Passagem, 93, Botafogo. 2. POR CARTA — Poderão ser solicitados exemplares atrasados também por carta, que deve ser enviada para DINAP — Distribuidora Nacional de Publicações — Números Atrasados — Estrada Velha de Osasco, 132, Jardim Teresa — CEP 06000 — Osasco — SP. Não envie pagamento antecipado. O atendimento será feito pelo reembolso postal e o pagamento, incluindo as despesas postais, deverá ser efetuado ao se retirar a encomenda na agência do Correio. 3. POR TELEX — Utilize o nº (011) 33 670 DNAP.
Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações, Lda. — Qta. Pau Varais, Azinhaga de Fetais — 2 685, Camarate — Lisboa; Apartado 57 — Telex 43 069 JARLIS P.
Atenção: Após seis meses do encerramento da coleção, os pedidos serão atendidos dependendo da disponibilidade do estoque.
Obs.: Quando pedir livros, mencione sempre título e/ou autor da obra, além do número da edição.

COLABORE CONOSCO

Encaminhe seus comentários, críticas, sugestões ou reclamações ao Serviço de Atendimento ao Leitor — Caixa Postal 9442, São Paulo — SP.

Editor
VICTOR CIVITA

REDAÇÃO
Diretora Editorial: Iara Rodrigues

Editor Executivo: Antonio José Filho

Editor Chefe: Paulo de Almeida
Editor de Texto: Cláudio A. V. Cavalcanti
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Ailton Oliveira Lopes, Dilvacy M. Santos, Grace Alonso Arruda, José Maria de Oliveira, Monica Lenardon Corradi
Secretária de Redação/Coordenadora: Stefania Crema
Secretários de Redação: Beatriz Hagström, José Benedito de Oliveira Damião, Maria de Lourdes Carvalho, Marisa Soares de Andrade, Mauro de Queiroz

COLABORADORES
Consultor Editorial Responsável: Dr. Renato M. E. Sabbatini (Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda., Campinas, SP
Tradução: Reinaldo Cúrcio
Tradução, adaptação, programação e redação: Abílio Pedro Neto, Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Raul Neder Porrelli
Coordenação Geral: Rejane Felizatti Sabbatini
Editora de Texto: Ana Lúcia B. de Lucena
Assistente de Arte: Dagmar Bastos Sampaio

COMERCIAL
Diretor Comercial: Roberto Martins Silveira
Gerente Comercial: Flávio Maculan
Gerente de Circulação: Denise Maria Mozol

PRODUÇÃO
Gerente de Produção: João Stungis
Coordenador de Impressão: Atílio Roberto Bonon

Preparador de Texto/Coordenador: Eliel Silveira Cunha
Preparadores de Texto: Alzira Moreira Braz, Ana Maria Dilguerian, Karina Ap. V. Grechi, Levon Yacubian, Luciano Tasca, Maria Teresa Galluzzi, Maria Teresa Martins Lopes, Paulo Felipe Mendrone
Revisor/Coordenador: José Maria de Assis
Revisoras: Conceição Aparecida Gabriel, Isabel Leite de Camargo, Ligia Aparecida Ricetto, Maria de Fátima Cardoso, Nair Lúcia de Brito
Paste-up: Anastase Potaris, Balduino F. Leite, Edson Donato

© Marshall Cavendish Limited, 1984/85.
© Editora Nova Cultural Ltda., São Paulo, Brasil, 1986.

Edição organizada pela Editora Nova Cultural Ltda.
Av. Brigadeiro Faria Lima, nº 2000 - 3º andar
CEP 01452 - São Paulo - SP - Brasil
(Artigo 15 da Lei 5 988, de 14/12/1973).
Esta obra foi composta na AM Produções Gráficas Ltda. e impressa na Divisão Gráfica da Editora Abril S.A.

SIMULAÇÕES ESPACIAIS

- ESCOLHA DA TRAJETÓRIA
- AVALIAÇÃO DA DISTÂNCIA
- CIRCUNDANDO A TERRA
- MANOBRAS NO ESPAÇO
- MOVIMENTO PLANETÁRIO

Atingir as estrelas é um antigo sonho do homem. Com nossos programas, que permitem simular a trajetória de um corpo no espaço, você estará mais perto da realização desse sonho.

No artigo sobre trajetórias (página 766), vimos como a velocidade de um projétil pode ser separada, para fins de análise, em uma componente vertical e outra horizontal. Vimos também como a distância do ponto de partida ao ponto de chegada pode mudar conforme o ângulo de lançamento e a velocidade inicial. Neste artigo, avançamos mais um pouco no estudo dos corpos em movimento. Analisamos objetos que se movem em campos de baixa gravidade, simulando sua trajetória a partir de pequenas distâncias da superfície da Terra e estudando corpos em órbita.

Digite este primeiro programa. Ele é um bom exemplo de como utilizar seu conhecimento de trajetórias para montar jogos que usam disparos ou outros lançamentos mais desafiadores.

```
10 FOR n=0 TO 31: READ a:
POKE USR "a"+n,a: NEXT n
```

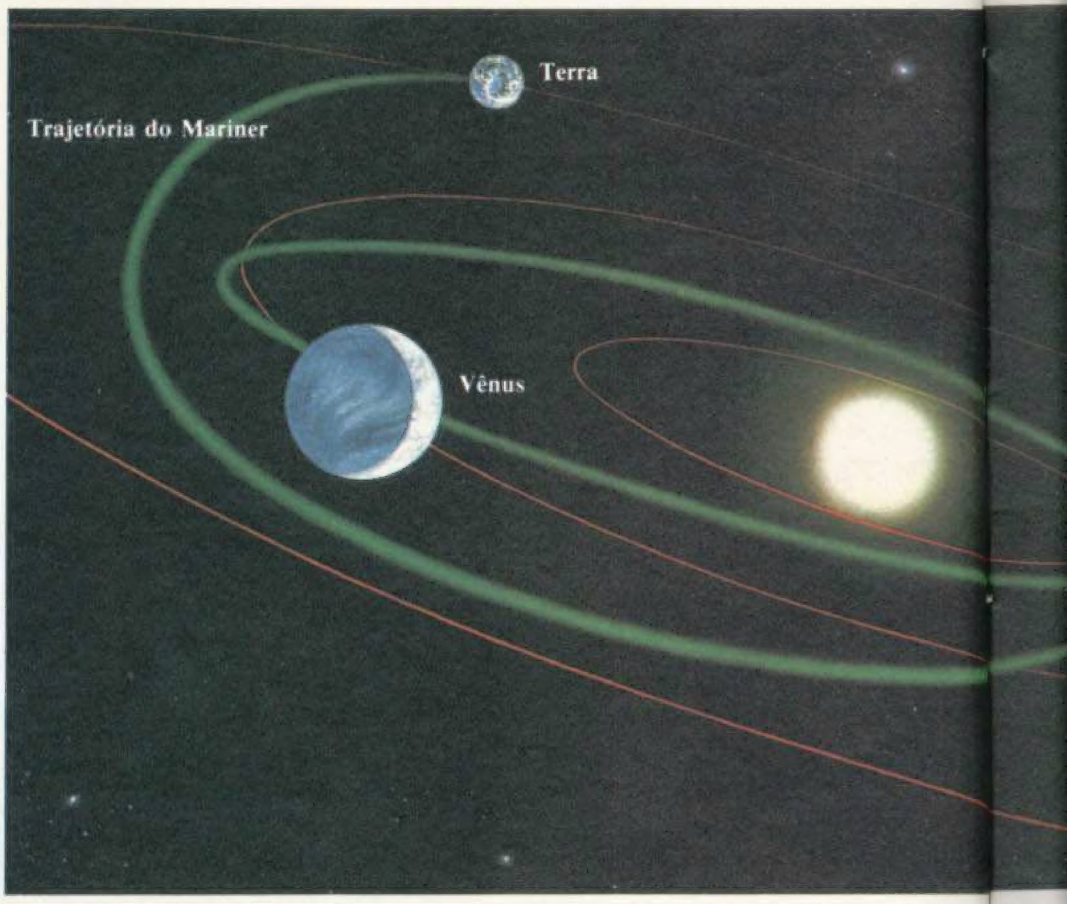
```
 20 BORDER 4: PAPER 0: INK 9:
 CLS
 70 LET a=INT (RND*5): LET b=
 INT (RND*5)+26: LET c=INT (
 RND*8)+2: LET h=INT (RND*8)+2
 : LET st=INT (RND*100-c*8)+1:
  LET d=0
 90 LET d=d+1
100 CLS : GOSUB 300
110 INPUT "ANGULO ?",a2
120 IF a2>89 OR a2<1 THEN
GOTO 110
130 INPUT "VELOCIDADE ?",e
140 IF e=0 THEN  GOTO 100
160 LET an=a2*(PI/180): LET
x=0
170 LET x2=x+(a+1)*8
180 LET y=8+(x*TAN an-4*x*x/(e
```

```
*e*COS an*COS an))
185 IF ATTR (21-INT (y/8),INT
(x2/8))=6 THEN  GOTO 245
190 IF y<=0 THEN  GOTO 245
200 IF (y>175 OR x2>255) AND d
<10 THEN  GOTO 90
205 IF y>175 OR x2>255 THEN
GOTO 270
210 PLOT  INK 8;x2,y: SOUND
.01,y/10
220 LET x=x+3
230 GOTO 170
245 IF x2>=b*8+3 AND x2<=b*8+
10 THEN  PRINT AT 21,b;CHR$
145: FOR n=20 TO 0 STEP -1:
SOUND .01,n: NEXT n: GOTO 270
246 IF d<10 THEN  GOTO 90
270 IF d=10 THEN  PRINT AT 8,
```

```
10;"ERROU !": GOTO 280
275 PRINT  INVERSE 1;AT 8,10;"
BOM TIRO !";AT 10,5;"VOCE CONS
EGUIU APOS ";d
280 PAUSE 100: PRINT  BRIGHT 1
; PAPER 2; INK 6;AT 13,8;"OUTR
 A VEZ?": PAUSE 200
290 GOTO 70
300 PRINT  INK 5;AT 21,a;CHR$
144;AT 21,b;CHR$ 146
310 FOR n=1 TO c: PRINT AT 21-
n+1,12;: FOR m=1 TO c: PRINT
INK 6;CHR$ 147;: NEXT m: NEXT
 n
320 RETURN
500 DATA 3,6,60,40,104,60,126,
255
510 DATA 36,90,165,90,60,155,
24,60
520 DATA 24,36,66,153,153,66,
36,127
530 DATA 28,42,85,170,127,170,
85,255
```

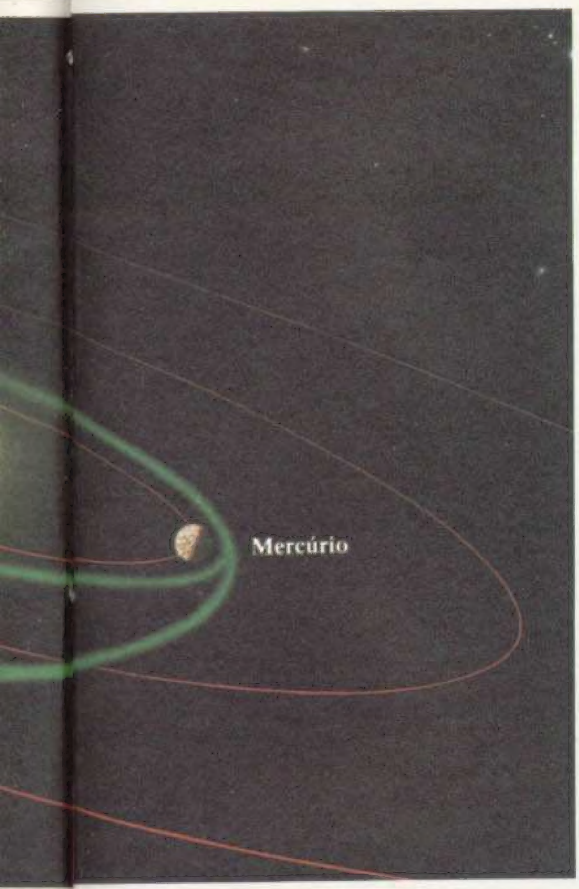
```
10 PMODE 4:DIM G(1),E(1),T(1),B
(1)
20 FOR K=1536 TO 2528 STEP 32:R
EAD A:POKE K,A:NEXT
30 GET (0,0)-(7,7),G,G:GET(0,8)
-(7,15),E,G
40 GET (0,16)-(7,23),T,G:GET(0,
24)-(7,31),B,G
50 DATA 3,6,60,40,104,60,126,25
5,36,90,165,90,60,155,24,60
60 DATA 24,36,66,153,153,66,36,
```

```
127,28,42,85,170,127,170,85,255
70 A=RND(51)-1:B=RND(51)+197:C=
RND(8):H=RND(8)+1:ST=RND(100-C*
8)+70:D=0:CLS
80 D=D+1
90 PCLS:SCREEN 1,1:GOSUB 320
100 IF INKEY$="" THEN 100
110 PRINT:INPUT " ANGULO ";A2
120 IF A2>89 OR A2<1 THEN 110
130 INPUT " VELOCIDADE ";E
140 IF E=0 THEN 130
150 SCREEN 1,1
160 AN=A2*ATN(1)/45:X=0
170 X2=X+A+8
180 Y=183-(X*TAN(AN)-4*X*X/(E*E
*COS(AN)*COS(AN)))
190 IF Y>190 THEN 250
200 IF X2>=ST AND X2<ST+C*8+7 A
ND Y>183-H*8 THEN 250
210 IF Y>0 THEN PSET(X2,Y,5):SO
UND 200-Y,1 ELSE SOUND 255 AND
(200-Y),1
220 X=X+3
230 IF X2<255 THEN 170
240 GOTO 290
250 IF Y>190 THEN Y=184
260 PUT(X2-4,Y)-(X2+3,Y+7),E,PS
ET:PLAY"T5001ADECBFGAEDBGDE"
270 PUT(A,184)-(A+7,191),G,PSET
:PUT(B,184)-(B+7,191),T,PSET
280 IF X2>=B AND X2<B+7 THEN 30
0
290 IF D<10 THEN 80
300 CLS:IF D=10 THEN PRINT @41,
" ERROU !" ELSE PRINT @41,"BOM
TIRO !":PRINT @164," VOCE CONSE
GUIU APOS";D;"DISPAROS."
310 FOR W=1 TO 3000:NEXT:PRINT:
PRINT" OUTRA VEZ ...":FOR W=1 T
O 1200:NEXT:GOTO 70
320 FOR X=0 TO C:FOR Y=0 TO H:P
UT(ST+X*8,184-8*Y)-(ST+X*8+7,19
1-8*Y),B,PSET
330 NEXT Y,X
340 PUT(A,184)-(A+7,191),G,PSET
:PUT(B,184)-(B+7,191),T,PSET:RE
TURN
```




```
10 W=RND(-TIME)
70 A=INT(RND(1)*51):B=INT(RND(1
)*51+198):C=INT(RND(1)*8+1):H=I
NT(RND(1)*9)+2:ST=INT(RND(1)*(1
01-C*8))+70:D=0:CLS
80 D=D+1
90 SCREEN2:GOSUB 320
100 IF INKEY$="" THEN 100
110 SCREEN0:PRINT:INPUT"Angulo
";A2
120 IF A2>89 OR A2<1 THEN 110
130 PRINT:INPUT"Velocidade ";E
140 IF E=0 THEN 130
150 SCREEN2:GOSUB 320
160 AN=A2*ATN(1)/45:X=0
170 X2=X+A+8
180 Y=183-(X*TAN(AN)-4*X*X/(E*E
*COS(AN)*COS(AN)))
190 IF Y>190 THEN 250
200 IF X2>=ST AND X2<ST+C*8+7 A
ND Y>183-H*8 THEN 250
210 IF Y>0 THEN PSET(X2,Y):A$="
N"+STR$(INT((190-Y)/2))+"L19":P
LAY A$
220 X=X+3
230 IF X2<255 THEN 170
240 GOTO 290
250 IF Y>190 THEN Y=184
260 LINE (X2,Y)-(X2,Y-8):PLAY "
O1AAA"
280 IF X2>=B AND X2<B+7 THEN 30
0
290 FOR W=1 TO 1500:NEXT:IF D<1
0 THEN 80
300 SCREEN0:IF D=10 THEN PRINT:
PRINT"TUDO FOI EM VÃO!" ELSE PR
INT:PRINT"BOM TIRO!":PRINT"Você
 acertou em";D;"disparos!"
310 FOR W=1 TO 3000:NEXT:PRINT:
PRINT"Outra vez...":FOR W=1 TO
1500:NEXT:GOTO 70
320 FOR X=0 TO C:FOR Y=0 TO H:P
SET (ST+X*8,184-Y*8)
330 NEXT Y,X
340 LINE (A,184)-(A+5,184):LINE
 (A+3,184)-(A+3,180):LINE (B,18
4)-(B+5,184):LINE (B+3,184)-(B+
3,180):RETURN
```

```
10  FOR I = 770 TO 795: READ A:
 POKE I,A: NEXT
20  POKE 768,2
70 A =  INT ( RND (1) * 51):B =
  INT ( RND (1) * 51 + 198):C =
  INT ( RND (1) * 8) + 1:H =  I
NT ( RND (1) * 9) + 2:ST =  INT
 ( RND (1) * (101 - C * 8)) + 7
0:D = 0: HOME : VTAB 21
80 D = D + 1
90  HGR : HCOLOR= 3: GOSUB 320
100  GET A$
110  INPUT "ANGULO? ";A2
120  IF A2 > 89 OR A2 < 1 THEN
110
130  VTAB  PEEK (37): HTAB 20:
INPUT "VELOCIDADE? ";E
140  IF E = 0 THEN 130
160 AN = A2 *  ATN (1) / 45:X =
 0
170 X2 = X + A + 8
180 Y = 157 - (X *  TAN (AN) -
4 * X * X / (E * E *  COS (AN)
*  COS (AN)))
190  IF Y > 160 THEN 250
200  IF (X2 >  = ST) AND (X2 <
ST + C * 8 + 7) AND (Y > 157 -
H * 8) THEN 250
210  IF Y > 0 THEN  HPLOT X2,Y:
 POKE 768,2: POKE 769,200 - Y:
CALL 770
220 X = X + 3
230  IF X2 < 259 THEN 170
240  GOTO 290
250  IF Y > 158 THEN Y = 158
260  HPLOT X2,Y TO X2,Y - 8: FO
R X = 1 TO 12:W =  PEEK ( - 163
36): NEXT
280  IF X2 >  = B AND X2 < B +
7 THEN 300
290  FOR I = 1 TO 1000: NEXT :
```




```
IF D < 10 THEN 80
300  FOR I = 1 TO 1500: NEXT :
HOME : TEXT : IF D = 10 THEN  P
RINT : PRINT "TUDO FOI INUTIL!"
: GOTO 310
305  PRINT : PRINT "BOM TIRO!":
 PRINT "VOCE ACERTOU EM ";D;" D
ISPAROS!"
310  FOR I = 1 TO 4000: NEXT :
PRINT : PRINT "OUTRA VEZ...": F
OR I = 1 TO 4000: NEXT : GOTO 7
0
320  FOR X = 0 TO C: FOR Y = 0
TO H: HPLOT ST + X * 8,158 - Y
* 8
330  NEXT : NEXT
340  HPLOT A,158 TO A + 5,158:
HPLOT A + 3,158 TO A + 3,150: H
PLOT B,158 TO B + 5,158: HPLOT
B + 3,158 TO B + 3,150: RETURN

5000  DATA  172,1,3,174,1,3,169
,4,32,168,252,173,48,192,232,20
8,253,136,208,239,206,0,3,208,2
31,96
```

O programa solicita que você especifique a velocidade e o ângulo de lançamento para executar um disparo de um ponto próximo ao canto inferior esquerdo da tela até um alvo situado junto ao canto inferior direito. Dois fatores dificultam o jogo: a determinação aleatória da distância entre o ponto de lançamento e o alvo e a presença de uma barreira de tamanho variável entre os dois pontos. Qualquer trajetória que se escolha deve ser alta o bastante para transpor essa barreira.

AVALIAÇÃO DA DISTÂNCIA

Conhecemos só as posições do atirador, do obstáculo e do alvo. No entanto, podemos estimar corretamente a velocidade e o ângulo necessários para produzir uma curva que leva o projétil acima do obstáculo até atingir o alvo. Com um pouco de prática, você verá que é possível obter grande proporção de acertos em uma série de tiros.

Mas, em geral, as coisas não são tão fáceis. Muitas vezes temos apenas uma idéia aproximada da distância até o alvo, e devemos calcular a velocidade e ângulo de tiro com este dado impreciso. Analisando se o tiro caiu para cá ou para lá do alvo, avaliações mais precisas vão sendo feitas, até que se consiga sucesso. O programa seguinte mostra esse processo. Depois de gravar o programa anterior, apague-o e digite estas linhas:

```
~10 CLS
 20 INPUT "VELOCIDADE INICIAL
 (1-10000 m/s)",sp
 30 IF sp<1 OR sp>10000 THEN
GOTO 20
 40 INPUT AT 4,0;"ANGULO (1-90
 graus)",a
 50 IF a<1 OR a>=90 THEN  GOTO
  40
 60 LET a=a*(PI/180)
 70 LET r=sp*sp*SIN (2*a)/10
 80 PRINT AT 10,3;"O ALCANCE F
 OI DE ";INT (r+.5);" metros"
 90 PRINT AT 20,1;"QUALQUER TE
 CLA PARA RECOMECAR    (0 SAI)"
100 PAUSE 0: LET a$=INKEY$: IF
a$="" THEN  GOTO 100
110 IF a$<>"0" THEN  GOTO 10
```

```
10 CLS
20 INPUT" VELOC. INICIAL (1-100
00 M/S)  ";SP
30 IF SP<1 OR SP>10000 THEN 10
40 INPUT" ANGULO (1-90 GRAUS)";
A
50 IF A<1 OR A>90 THEN 40
60 A=A*ATN(1)/45
70 R=SP*SP*SIN(2*A)/10
80 PRINT:PRINT" O ALCANCE FOI D
E";INT(R+.5);"METROS"
90 PRINT:PRINT" QUALQUER TECLA
PARA RECOMECAR    '0' SAI"
100 A$=INKEY$:IF A$="" THEN 100
110 IF A$="0" THEN CLS:END ELSE
 10
```

```
10 CLS
20 INPUT"VELOCIDADE DE LANÇAMEN
TO (1-10000 M/S)";SP
30 IF SP<1 OR SP>10000 THEN 10
40 INPUT"ANGULO DE LANÇAMENTO (
1-90 GRAUS)";A
50 IF A<1 OR A>89 THEN 40
60 A=A*ATN(1)/45
70 R=SP*SP*SIN(2*A)/10
80 PRINT:PRINT"A DISTANCIA ALCA
NÇADA É";INT(R+.5);"METROS"
90 PRINT:PRINT"PRESSIONE <0> PA
RA TERMINAR OU QUALQUER      OUT
RA TECLA PARA CONTINUAR";
100 A$=INKEY$:IF A$="" THEN 100
110 IF A$="0" THEN CLS:END ELSE
 10
```

```
10  HOME
20  INPUT "VELOCIDADE DE LANCAM
ENTO (1-1000 M/S)?  ";SP
30  IF SP < 1 OR SP > 10000 THE
N 10
40  INPUT "ANGULO DE LANCAMENTO
 (1-90 GRAUS)";A
50  IF A < 1 OR A > 89 THEN 40
60  A = A *  ATN (1) / 45
70  R = SP * SP *  SIN (2 * A) /
 10
80  PRINT : PRINT "A DISTANCIA
ALCANCADA E DE "; INT (R + .5);
" METROS"
90  PRINT : PRINT "TECLE <0> PA
RA TERMINAR OU QUALQUER
OUTRA TECLA PARA CONTINUAR";
100  GET A$
110  IF A$ = "0" THEN  HOME : E
ND
120  GOTO 10
```

O programa pede valores para a velocidade inicial (linha 20) e para o ângulo de lançamento (linha 40). A linha 70 calcula e mostra a distância a que esses dois valores levariam o projétil. A variável **R** corresponde à distância; **SP** refere-se à velocidade e **A**, ao ângulo. O valor aproximado para a aceleração da gravidade próximo à superfície da Terra é igual a 10.

Ignora-se o efeito da resistência do ar mas, em experimentos reais, ele deveria ser considerado. A maior velocidade de lançamento permitida é 10000 m/s. A velocidade de 2000 m/s foi usada em lançamentos de objetos a grandes altitudes, para pesquisar o espaço. Os objetos atingem 180 km de altitude quando lançados a um ângulo de 90°. Mas, sem contar a resistência do ar, chegam a 250 km.

CIRCUNDANDO A TERRA

Se um objeto é lançado em um ângulo que o leva muito longe da superfície da Terra, o efeito da fricção do ar diminui. Porém, quando se pretende que ele atinja uma grande distância do ponto de lançamento, é preciso levar em conta a curvatura da Terra.

Um dos primeiros a considerar esta questão foi o cientista inglês Isaac Newton. Ele imaginou um poderoso canhão que lançasse seus projéteis do alto de uma montanha suficientemente grande para que seu pico ficasse fora da atmosfera terrestre. Tiros dados a velocidades crescentes levariam a distâncias também crescentes se a Terra fosse plana. Mas, como ela é curva, a superfície se afasta do ponto de lançamento, fazendo com que o projétil percorra distâncias ainda maiores.

Eventualmente, argumentava Newton, um lançamento poderia ter uma velocidade inicial tão alta que o projétil nunca atingiria o solo, mas poderia, em termos teóricos, atingir o pesquisador bem na nuca. À medida que o projétil caísse em direção ao solo, este "cairia" sob ele (visto que a Terra é redonda). Dessa maneira, o projétil ficaria para sempre em queda livre — ou seja, estaria em órbita.

Uma vez que um objeto escapa da gravidade de um planeta, sua velocidade e distância determinam um tipo de órbita — circular ou elíptica.

Para que se possa prever a trajetória e fazer medições relacionadas a um planeta ou satélite, sua órbita precisa ser conhecida, ou seja, devemos saber a forma exata da elipse. O grau de "achatamento" da elipse é sua excentricidade (**E**), que corresponde à proporção entre seu comprimento e sua largura. Quando **E** é igual a 1, temos uma elipse tão larga quanto comprida, ou seja, um círculo.

Digite e execute este programa para ver o efeito da variação de **E** entre valores menores e maiores que 1:

```
10 CLS
30 INPUT "EXCENTRICIDADE (.1
a 1.9)",e
40 IF e<.1 OR e>1.9 THEN
GOTO 30
50 PLOT 127,87+e*40
60 FOR a=0 TO 2*PI+.2 STEP .1
70 DRAW 127+(40*SIN a)-PEEK
23677,87+(e*40*COS a)-PEEK
23678
80 NEXT a
90 PRINT AT 20,1;"QUALQUER TE
CLA PARA RECOMECAR   (0 SAI)"
100 PAUSE 0: LET a$=INKEY$:
IF a$="" THEN  GOTO 100
110 IF a$<>"0" THEN  GOTO 10
```

```
10 PMODE 4,1:PCLS
30 CLS:INPUT"EXCENTRICIDADE (.1
 A 1.9) ";E
40 IF E<.1 OR E>1.9 THEN 30
50 SCREEN 1,1
70 CIRCLE(128,96),48,5,E
100 A$=INKEY$:IF A$="" THEN 100
110 IF A$="0" THEN CLS:END ELSE
 30
```

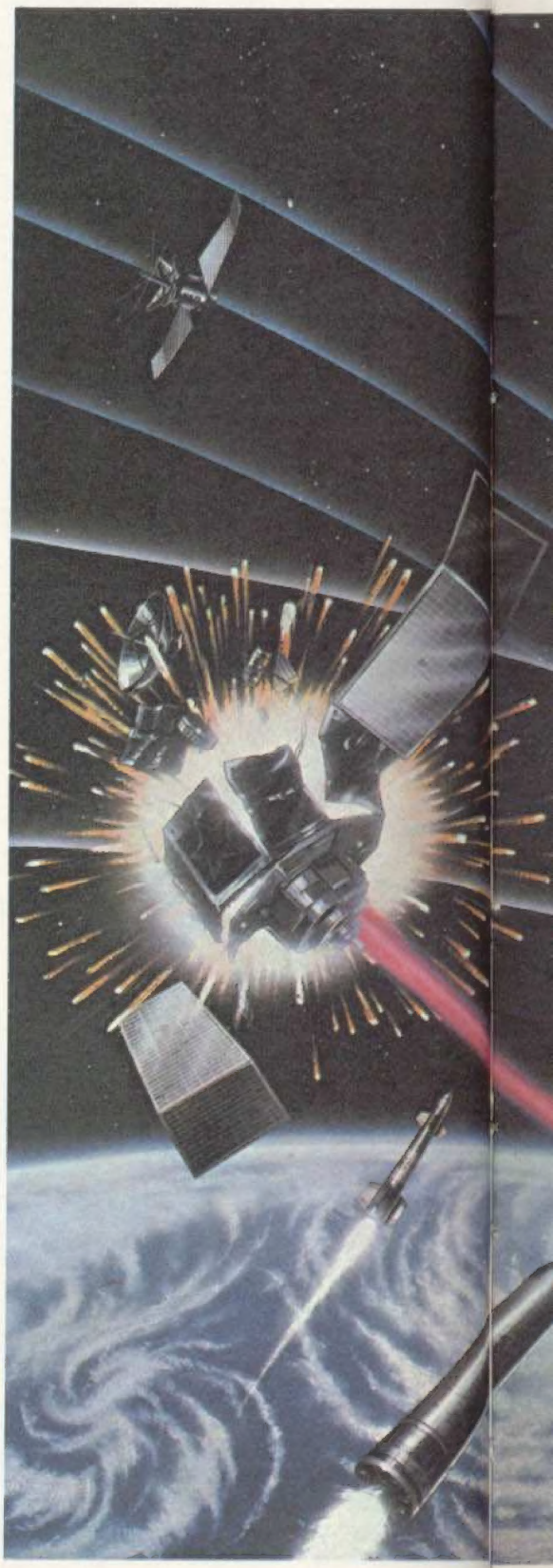
```
30 CLS:INPUT"EXCENTRICIDADE (0."
1 A 1.9)";E
40 IF E<.1 OR E>1.9 THEN 10
50 SCREEN 2:COLOR 15,4,4
70 CIRCLE(128,96),48,15,,,E
100 A$=INKEY$:IF A$="" THEN 100
110 IF A$="0" THEN CLS:END ELSE
 30
```

```
10  TEXT
30  HOME : INPUT "EXCENTRICIDAD
E (0.1 A 1.9) ";E
40  IF E < .1 OR E > 1.9 THEN 3
0
50  HGR2 : HCOLOR= 3
60 X = 130:Y = 90
70  FOR A = 0 TO 2 * 3.1416 STE
P .05
80  HPLOT X + 45 *  SIN (A),Y -
 (E * 45 *  COS (A))
90  NEXT
100  GET A$
110  IF A$ = "0" THEN  TEXT : H
OME : END
120  GOTO 10
```

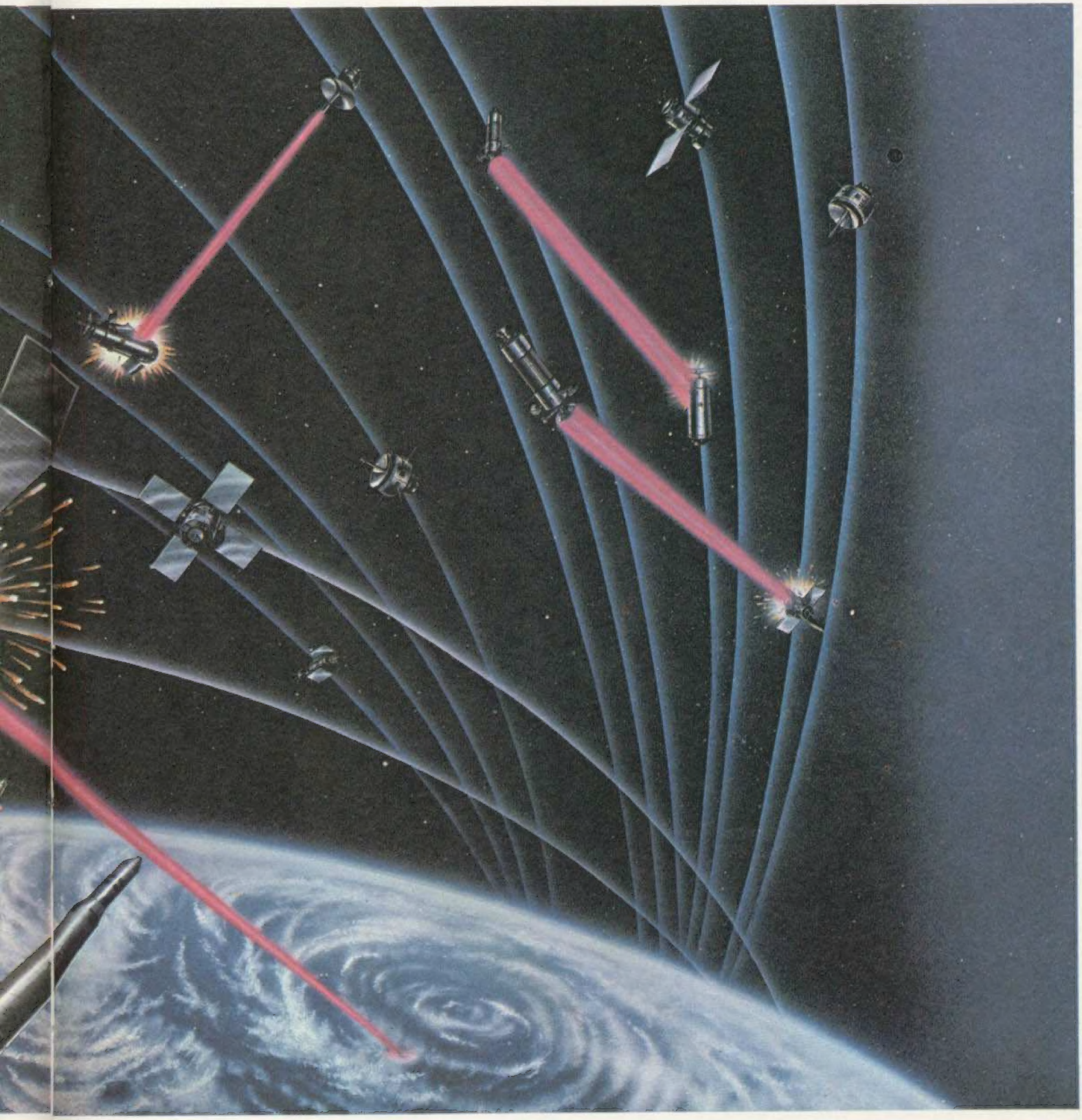
O programa gira entre as linhas 30 e 110 para desenhar elipses. A linha 30 determina o valor de **E** de acordo com sua escolha. O TRS-Color e o MSX usam o comando **CIRCLE** (linha 70) para desenhar as curvas. Os outros micros empregam um laço **FOR...NEXT** para desenhar cada ponto.

Digite valores de **E** na faixa indicada na tela. **E** igual a 1 corresponderá a um círculo; **E** menor que 1, a uma elipse achatada em cima e embaixo, e **E** maior que 1, a uma elipse achatada lateralmente. Assim, toda órbita pode ser considerada uma elipse, com um valor particular de **E**.

Uma vez em órbita, um satélite ou uma espaçonave não precisam de propulsão, pois estarão caindo livremente. O uso de foguetes irá movê-los para uma órbita diferente. Quanto menor for o raio da órbita, maior será a velocidade com que o objeto deve se mover. Digite o próximo programa para ver como isso funciona:



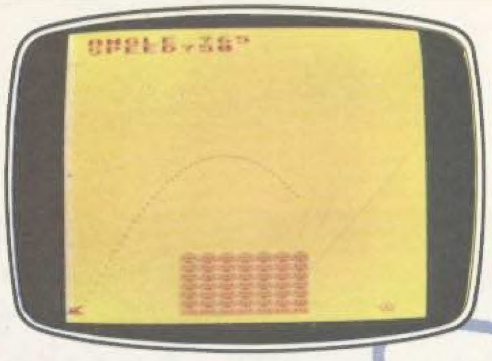
```
10 CLS : LET gc=0
40 LET r=40: LET rt=10+INT (
RND*60): IF ABS (r-rt)<10
THEN  GOTO 40
50 CIRCLE 127,87,4
60 LET s=.1: LET a=0: LET at=
INT (RND*10)+1: LET f=0
 80 LET a=a+s
100 LET at=at+.1*SQR ((40/rt)^
3)
110 IF INKEY$="7" AND r<85
THEN  LET f=f+1: LET r=r+2:
```



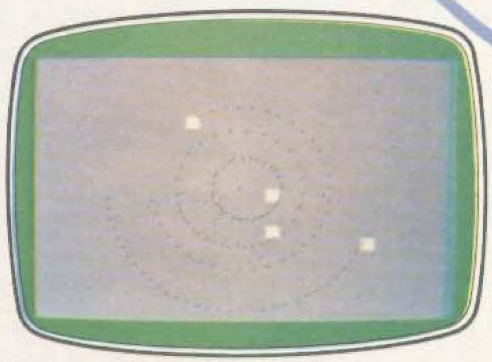
```
LET s=s*SQR (((r+2)/r)^3):
GOTO  130
115 IF INKEY$="6" AND r>8 THEN
 LET f=f+1: LET r=r-2: LET s=s
*SQR (((r+2)/r)^3): GOTO 130
120 FOR p=1 TO 5: NEXT p
130 LET x=r*SIN a: LET y=r*COS
a
140 LET xt=rt*SIN at: LET yt=r
t*COS at
150 PLOT 127+x,87+y
160 SOUND .02,r/2
165 IF gc<>0 THEN  PLOT  OVER
1;127+ox,87+oy
170 PLOT  OVER 1;127+xt,87+yt:
LET ox=xt: LET oy=yt: LET gc=1
180 IF ABS (x-xt)>4 OR ABS (y-
yt)>4 THEN  GOTO 80
190 PRINT AT 20,12;f;" PONTOS"
200 GOTO 200
```

```
10 PMODE 4:PCLS:SCREEN 1,1
40 R=30:RT=10+RND(70):IF ABS(R-
RT)<20 THEN 40
50 DRAW"BM128,96S8NUNRNDL"
60 S=.1:A=0:AT=RND(10):F=0
80 A=A+S
95 LT=AT
100 AT=AT+.1*SQR((40/RT)^3)
110 IF PEEK(341)=247 AND R<95 T
HEN F=F+1:R=R+1:S=S*SQR(((R-1)/
R)^3)ELSE IF PEEK(342)=247 AND
R>8 THEN F=F+1:R=R-1:S=S*SQR(((
R+1)/R)^3)ELSE FOR K=1 TO 25:NE
XT
130 X=R*SIN(A):Y=R*COS(A)
140 XT=RT*SIN(AT):YT=RT*COS(AT)
150 PSET(128+X,96-Y,5)
160 SOUND R*2,1
170 PSET(128+XT,96-YT,5)
175 PSET(128+RT*SIN(LT),96-RT*C
OS(LT),0)
180 IF ABS(X-XT)>=4 OR ABS(Y-YT
)>=4 THEN 80
190 CLS:PRINT" VOCE USOU";F;"TO
QUES"
```

```
5 W=RND(-TIME)
10 SCREEN2
40 R=30:RT=INT(RND(1)*70)+11:IF
 ABS(R-RT)<20 THEN 40
50 CIRCLE (128,96),2
60 S=.1:A=0:AT=INT(RND(1)*11)+1
:F=0
80 A=A+S
95 LT=AT
100 AT=AT+.1*SQR((40/RT)^3)
105 FOR I=1 TO 5:A$=INKEY$:IF A
$<>"" THEN I=5
234 NEXT
110 IF A$=CHR$(30) AND R<95 THE
N F=F+1:R=R+1:S=S*SQR(((R-1)/R)
^3)
120 IF A$=CHR$(31) AND R>8 THEN
 F=F+1:R=R-1:S=S*SQR(((R+1)/R)^
3)
```



Um jogo simples usando trajetórias.



Órbita de quatro planetas internos.

```
130 X=R*SIN(A):Y=R*COS(A)
140 XT=RT*SIN(AT):YT=RT*COS(AT)
150 PSET (128+X,96-Y)
170 PSET (128+XT,96-YT)
175 PSET (128+RT*SIN(LT),96-RT*
COS(LT)),0
180 IF ABS(X-XT)>=4 OR ABS(Y-YT
)>=4 THEN 80
190 SCREEN0:PRINT"VOCE USOU";F;
"TOQUES"
```

```
10  HGR2
40 R = 30:RT =  INT ( RND (1) *
 70) + 11: IF  ABS (R - RT) < 2
0 THEN 40
50  HPLOT 126,96 TO 130,96: HPL
OT 128,94 TO 128,98
60 S = .1:A = 0:AS =  INT ( RND
 (1) * 11):F = 0
80 A = A + S
95 LT = AS
100 AS = AS + .1 *  SQR ((40 /
RT) ^ 3)
105  POKE  - 16368,0
110  IF  PEEK ( - 16384) = 181
AND R < 95 THEN F = F + 1:R = R
 + 1:S = S *  SQR (((R - 1) / R
) ^ 3)
120  IF  PEEK ( - 16384) = 182
AND R > 8 THEN F = F + 1:R = R
- 1:S = S *  SQR (((R + 1) / R)
 ^ 3)
125  POKE  - 16368,0
130 X = R *  SIN (A):Y = R *  C
OS (A)
140 XT = RT *  SIN (AS):YT = RT
 *  COS (AS)
150  HPLOT 128 + X,96 - Y
170  HPLOT 128 + XT,96 - YT
175  HCOLOR= 0: HPLOT 128 + RT
*  SIN (LT),96 - RT *  COS (LT)
: HCOLOR= 3
180  IF  ABS (X - XT) >  = 4 OR
  ABS (Y - YT) >  = 4 THEN 80
190  TEXT : HOME : PRINT "VOCE
USOU ";F;" TOQUES"
```

MANOBRAS NO ESPAÇO

Executando o programa, você verá um satélite-alvo e a trajetória de uma nave em órbita. Tente igualar as duas órbitas usando as setas para cima e para baixo (elas correspondem às teclas 6 e 7 no Spectrum e no Apple).

A linha 40 determina o raio R da órbita da nave como 200; o raio do alvo é definido ao acaso. A linha 60 especifica as variáveis para as posições iniciais. O ponto essencial do programa está na linha 100, que usa uma importante lei da física: o quadrado do tempo para percorrer uma órbita completa dividido pelo cubo do raio é uma constante. Por essa razão, o SQR e a potência 3 estão na linha 100.

Faça as manobras utilizando as teclas citadas para aumentar ou diminuir a órbita. Lembre-se de que quanto menor a órbita, maior a rapidez.

MOVIMENTO PLANETÁRIO

Na realidade, manobrar de uma órbita para outra é muito mais complicado do que sugere o programa anterior. A dificuldade não se encontra na mudança de uma órbita para outra, mas na localização de um alvo na vastidão do espaço. E, para complicar um pouco mais, é preciso levar em consideração que a gravidade do Sol, da Lua e, também, dos planetas tem efeito sobre o movimento de uma espaçonave.

Na prática, poderosos computadores são as ferramentas dos navegantes do espaço. Eles controlam a velocidade, o tempo e a direção dos foguetes, garantindo que estes se mantenham na trajetória desejada.

Uma vez na órbita correta, a nave cai livremente no campo gravitacional do sistema solar. Ela precisa apenas de pequenas correções de curso no decorrer de meses e até anos de viagem — é quase tão previsível quanto os planetas que circundam o Sol.

O programa seguinte permitirá a observação do movimento dos planetas.

```
10 BORDER 4: CLS : LET gc=0
20 DIM d(9): DIM p(9): DIM i(
9): DIM a(9): DIM b(9): DIM x
(9): DIM y(9)
30 FOR t=1 TO 9: READ d(t),p(
t): NEXT t
40 INPUT "Quantos planetas (1
-9) ?",s
50 IF s<1 OR s>9 THEN  GOTO
 40
60 LET sc=d(s)/325: LET t=p(s
)/75
70 PRINT "Ha um espaco de ";
INT t;" dias"'"entre cada pon
to.": PRINT #1;" <SPACE> PARA
 CONTINUAR"
75 IF INKEY$<>CHR$ 32 THEN
GOTO 75
80 CLS
100 PRINT #1;"PRESSIONE <SPACE
> PARA RECOMECAR"
110 LET n=1: LET m=1: LET q=PI
/180
120 FOR q=1 TO s
130 LET r=d(q)/sc: LET a=r:
LET b=r: LET e=0: LET p=p(q)/t
140 IF q=1 THEN  LET e=.2: LET
b=a*.98
150 IF q=8 THEN  LET e=.26:
LET b=a*.96
160 LET i(q)=i(q)+360/p
180 LET y=g*i(q): LET x=INT (a
*(COS y-e)): LET y=INT (b*SIN
y)
200 IF gc<>0 THEN  PLOT
BRIGHT 0;127+x(q),87+y(q)
210 PLOT  BRIGHT 1;127+a(q)/4,
87+b(q)/4: LET x(q)=a(q)/4:
LET y(q)=b(q)/4
240 LET a(q)=x: LET b(q)=y:
NEXT q
250 LET m=m+t
260 IF INKEY$=CHR$ 32 THEN
RUN
270 LET gc=1: GOTO 120
280 DATA 58,88,108,225,150,365
,228,687
290 DATA 778,4333,1427,10759,
2870,30685
300 DATA 4497,60190,5969,90741
```

```
10 PMODE 4:PCLS
20 DIM D(8),P(8),I(8),A(8),B(8)
30 FOR T=0 TO 8:READ D(T),P(T):
NEXT
40 CLS:INPUT " QUANTOS PLANETAS
 (1-9) ";S
50 IF S<1 OR S>9 THEN 40
60 S=S-1:SC=D(S)/90:T=P(S)/75
70 PRINT:PRINT" HA UM ESPACO DE
";INT(T);"DIAS":PRINT" ENTRE CA
DA PONTO":PRINT:PRINT:PRINT "PR
ESSIONE<ESPACO>PARA CONTINUAR"
75 IF INKEY$<>" " THEN 75
80 SCREEN 1,1
90 CIRCLE(128,96),1,5
110 G=ATN(1)/45
120 FOR Q=0 TO S
130 R=D(Q)/SC:A=R:B=R:E=0:P=P(Q
)/T
140 IF Q=0 THEN E=.2:B=A*.98
150 IF Q=8 THEN E=.26:B=A*.96
155 IF S>5 AND Q<6 THEN 245
160 IF P>3 THEN P=INT(P+.5)
170 I(Q)=I(Q)+360/P
180 Y=G*I(Q):X=INT(A*(COS(Y)-E)
):Y=INT(B*SIN(Y))
200 CIRCLE(128+A(Q),96-B(Q)),1,
0
220 PSET (128+A(Q),96-B(Q),5)
230 CIRCLE(128+X,96-Y),1,5
240 A(Q)=X:B(Q)=Y
245 NEXT
250 M=M+T
260 IF INKEY$=" " THEN RUN
270 GOTO 120
280 DATA 58,88,108,225,150,365,
228,687
290 DATA 778,4333,1427,10759,28
70,30685
300 DATA 4497,60190,5969,90741
```

```
10 CLS
20 DIM D(8),P(8),I(8),A(8),B(8)
30 FOR T=0 TO 8:READ D(T),P(T):
NEXT
40 INPUT"QUANTOS PLANETAS (1-9)
";S
50 IF S<1 OR S>9 THEN 40
60 S=S-1:SC=D(S)/90:T=P(S)/75
70 PRINT:PRINT"HA UM ATRASO DE"
;INT(T);"DIAS ENTRE CADA PONTO"
:PRINT:PRINT:PRINT"TECLE A BARR
A DE ESPAÇOS PARA CONTINUAR"
75 IF INKEY$<>" " THEN 75
80 SCREEN 2
90 CIRCLE (128,96),1,15
100 G=ATN(1)/45
120 FOR Q=0 TO S
130 R=D(Q)/SC:A=R:B=R:E=0:P=P(Q
)/T
140 IF Q=0 THEN E=.2:B=A*.98
150 IF Q=8 THEN E=.26:B=A*.96
160 IF P>3 THEN P=INT(P+.5)
170 I(Q)=I(Q)+360/P
180 Y=G*I(Q):X=INT(A*(COS(Y)-E)
):Y=INT(B*SIN(Y))
200 CIRCLE(128+A(Q),96-B(Q)),1,
15
220 PSET(128+A(Q),96-B(Q)),5
230 CIRCLE(128+X,96-Y),1,6
240 A(Q)=X:B(Q)=Y
245 NEXT
250 M=M+T
260 IF INKEY$=" " THEN RUN
270 GOTO 120
280 DATA 58,88,108,225,150,365,
228,687
290 DATA 778,4333,1427,10759,28
70,30685
300 DATA 4497,60190,5969,90741
```

```
10  TEXT
20  DIM D(8),P(8),I(8),A(8),B(8
)
30  FOR T = 0 TO 8: READ D(T),P
(T): NEXT
40  HOME : INPUT "QUANTOS PLANE
TAS (1-9) ";S
50  IF S < 1 OR S > 9 THEN 40
60 S = S - 1:SC = D(S) / 90:T =
 P(S) / 75
70  PRINT : PRINT "HA UM ATRASO
 DE "; INT (T);" DIAS ENTRE CAD
A PONTO": PRINT : PRINT "TECLE
 A BARRA DE ESPACOS PARA CONTINU
AR"
75  GET A$: IF A$ <  >  CHR$ (3
2) THEN 75
80  HGR2 : HCOLOR= 3
90  HPLOT 128,96
110 G =  ATN (1) / 45
120  FOR Q = 0 TO S
130 R = D(Q) / SC:A = R:B = R:E
 = 0:P = P(Q) / T
140  IF Q = 0 THEN E = .2:B = A
 * .98
150  IF Q = 8 THEN E = .26:B =
A * .96
155  IF S > 5 AND Q < 6 THEN 24
5
160  IF P > 3 THEN P =  INT (P
+ .5)
170 I(Q) = I(Q) + 360 / P
180 Y = G * I(Q):X =  INT (A *
( COS (Y) - E)):Y =  INT (B *
SIN (Y))
200  HCOLOR= 0: HPLOT 129 + A(Q
),96 - B(Q) TO 128 + A(Q),95 -
B(Q) TO 127 + A(Q),96 - B(Q) TO
 128 + A(Q),97 - B(Q)
210  HCOLOR= 3: HPLOT 128 + A(Q
),96 - B(Q)
220  HPLOT 129 + X,96 - Y TO 12
8 + X,95 - Y TO 127 + X,96 - Y
TO 128 + X,97 - Y
240 A(Q) = X:B(Q) = Y
245  NEXT
250 M = M + T
260  POKE  - 16368,0
265  IF  PEEK ( - 16384) > 128
THEN  RUN
270  GOTO 120
280  DATA  55,88,108,225,150,36
5,228,687
290  DATA  778,4333,1427,10759,
2870,30685
300  DATA  4497,60190,5969,9074
1
```

Ao executar o programa, você deve escolher o número de planetas que deseja ver. Quanto maior o número deles, mais complicada a figura. Para identificar os planetas, lembre-se de que Mercúrio está mais perto do Sol, seguido por Vênus, Terra, Marte, Júpiter, Saturno, Urano, Netuno e Plutão.

Execute o programa com diferentes valores. Note que as órbitas de dois planetas — Mercúrio e Plutão — são elípticas. Na verdade, as outras também o são, mas têm uma excentricidade tão pequena que parecem circulares.

AVALANCHE: EFEITOS SONOROS

Vamos ouvir um som antes de colocar nosso personagem em cena. Que tal um grande sucesso de 1560? Sente-se diante de seu micro e prepare uma abertura musical digna de Avalanche.

Um jogo de ação não seria completo sem uma abertura musical. Nosso videogame toca a melocia *Greensleeves*, no tom e ritmo corretos. Talvez você critique nossa escolha, por julgá-la pouco adequada a um videogame. Contudo, de acordo com a lenda, Henrique VIII escreveu essa melodia para sua esposa Ana Bolena. Mais tarde, ela teve um fim bastante trágico, assim como será o de Willie, se cometer qualquer descuido no jogo. Além disso, não teremos que pagar direitos autorais — o que é muito importante...

A rotina Assembly listada a seguir utiliza a sub-rotina da ROM que controla o comando **SOUND**, o que facilita a execução das notas musicais.

```
 10 REM org 60000
 20 REM ld ix,57359
 30 REM msk ld b,19
 40 REM tune push bc
 50 REM ld d,(ix+1)
 60 REM ld e,(ix+0)
 70 REM ld h,(ix+3)
 80 REM ld l,(ix+2)
 90 REM push ix
100 REM call 949
110 REM pop ix
120 REM ld de,4
130 REM add ix,de
140 REM pop bc
150 REM djnz tune
160 REM ret
```

Podemos escolher qualquer melodia, contanto que forneçamos a essa rotina as informações necessárias sobre o que vai tocar. O programa BASIC que apresentamos a seguir coloca na memória os dados que o computador precisa para executar *Greensleeves*. As notas, com suas respectivas durações, formarão uma espécie de tabela, colocada logo abaixo da tabela ASCII que contém os textos do título e das instruções.

```
 5 CLEAR 57358
10 FOR n=57359 TO 57434 STEP
2: READ a: POKE n+1,INT (a/
256): POKE n,a-(256*INT (a/
256)): NEXT n
20 DATA 98,1460,233,1223,131,
1086,220,964,78,908,147,964,
261,1086,110,1297,131,1642,49
,1460,110,1297,233,1223,98,
1460,147,1460,44,1642,98,1460
,220,1297,92,1548,220,1959
```

COMO FUNCIONA

A rotina começa colocando o endereço 57359 no registro IX. Como registro-índice, IX vai servir de apontador e seu valor inicial corresponde ao início da tabela de notas.

Em seguida, 19 é colocado em B, registro que servirá como contador. Seu conteúdo inicial corresponde ao número de notas que serão executadas (o trecho de *Greensleeves* que vamos tocar tem dezenove notas). Esse valor é guardado temporariamente na pilha, pelo comando **push bc**. O registro C precisa ser guardado junto com o B porque não existe um comando que guarde registros de oito bits na pilha. As instruções **push** e **pop** só aceitam pares de registros como operandos.

Mas por que guardar o valor de B e C na pilha, se nenhum dos dois registros será usado até que a instrução **pop bc** recupere seu conteúdo original? Esta é uma boa pergunta. Enquanto o valor de BC está na pilha, o programa chama uma sub-rotina da ROM, que pode muito bem usar o registro B. Embora uma sub-rotina possa alterar o valor de qualquer registro, ela sempre deixa a pilha intacta. Assim, sempre que chamamos uma sub-rotina que não conhecemos bem, devemos guardar o conteúdo dos registros importantes na pilha.

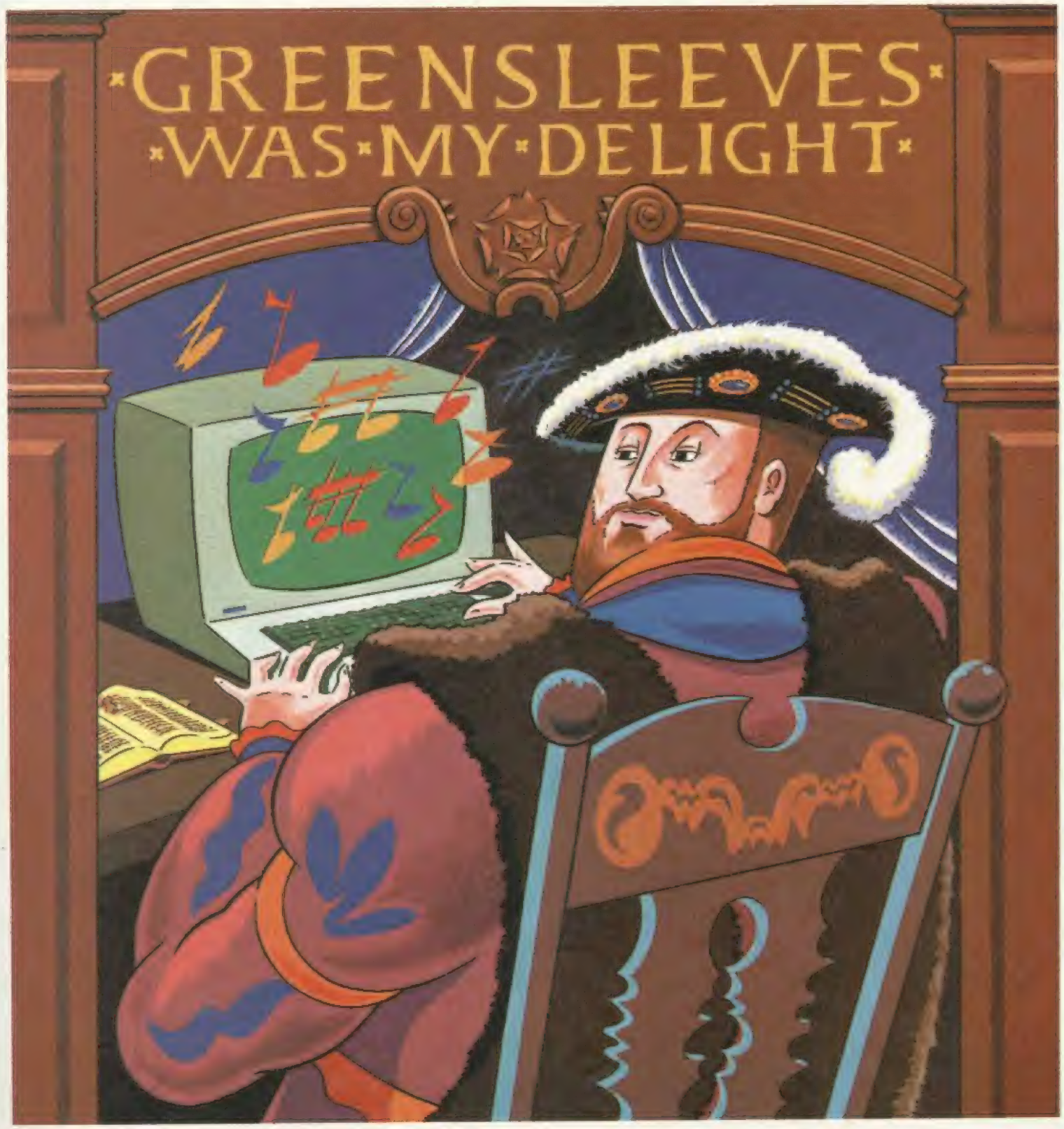
O primeiro número da tabela é colocado em DE; o segundo, em HL. A operação, diferentemente do usual, emprega comandos com endereçamento indexado, de modo que IX é usado como apontador. Os registros têm que ser carregados um de cada vez, já que não há comandos que carreguem pares de registros nesse tipo de endereçamento. Se consultarmos a listagem do programa BASIC, veremos que cada número ocupa dois bytes, ainda que muitos sejam menores que 256.

Os números formam pares. O primeiro valor determina a duração da nota — na realidade, ele especifica o número de ciclos gerados pela rotina do comando **SOUND**. O número corresponde, então, à freqüência da nota multiplicada pela sua duração em segundos. O segundo número determina a tonalidade da nota. Para calcular seu valor, podemos usar a fórmula 437500/f-30125, onde f é a freqüência da nota.

A rotina coloca o conteúdo de IX na pilha, para preservar o valor do apontador, e, em seguida, chama a sub-rotina da ROM. A sub-rotina do comando **SOUND** é chamada pela instrução **call 949**. Ela usa os valores de HL e DE para produzir o som. A instrução **pop ix** recupera da pilha a posição do apontador na tabela de notas; o ponteiro avança um par de notas, com a colocação de 4 em DE (**ld de, 4**) e a soma desse valor a IX (**add ix,de**). O emprego de **add** é necessário, já que não existe uma instrução especial para somar valores ao índice IX. Poderíamos usar quatro instruções **inc ix**, mas o processo seria mais demorado.

O valor original do contador volta para BC. A instrução **djnz** subtrai uma unidade de B e, se este não foi ainda re-

- A ESCOLHA DA MÚSICA
- CONTAGEM DAS NOTAS
- CÁLCULO DAS FREQÜÊNCIAS
- ROTINA PRINCIPAL E SUB-ROTINAS
- ARMAZENAGEM DA MÚSICA
- COMO MODIFICAR A MELODIA
- COMO SINCRONIZAR O SOM E A AÇÃO DO JOGO



duzido a zero, ela faz o processador voltar para emitir mais uma nota. Após dezenove notas, o processador deixa o laço e encontra **ret**.

Para testar a sub-rotina, utilize **RAND USR 60000**, mas não se esqueça de que a tabela de notas deve estar registrada na memória.

TOQUE SUA PRÓPRIA MELODIA

Mesmo depois de toda aquela conversa sobre Henrique VIII, talvez você queira ouvir outra música. Embora não seja aconselhável fazer a substituição dentro do nosso jogo, você pode usar a rotina separadamente.

Escolha a música de sua preferência e traduza-a nos parâmetros que a rotina utiliza para emitir notas musicais. Lembre-se, contudo, de que melodias diferentes requerem um outro valor para o contador B, que determina o número de notas executadas.

O TRS-Color executa a melodia *Greensleeves* por intermédio da rotina Assembly, que é dada a seguir. Não se esqueça de aumentar a memória disponível com **POKE 25,6:POKE 26,1:NEW** e também de proteger uma área para o programa em código.

```
10  ORG 30000
20  LDX #MUSIC
30  STX MUPOINT
40  LDA #19
50  PSHS A
60  LDA $FF01
70  ANDA #247
80  STA $FF01
90  LDA $FF03
100  ANDA #247
110  STA $FF03
120  LDA $FF23
130  ORA #8
140  STA $FF23
150  MAIN LDU MUPOINT
160  ORCC #$50
170  PULU A,X
180  CMPU #MUSIC+57
190  BLO MONE
200  LDU #MUSIC
210  MONE STU MUPOINT
220  PSHS X
230  LDB #252
240  MTWO STB $FF20
250  MTHR LEAX -1,X
260  BNE MTHR
270  LDX ,S
280  CLR $FF20
290  MFOU LEAX -1,X
300  BNE MFOU
310  LDX ,S
320  DECA
330  BNE MTWO
340  LEAS 2,S
350  DEC ,S
360  BNE MAIN
370  ANDCC #$AF
380  PULS A,PC
390  MUPOINT FDB $758A
400  MUSIC FCB 98,0,189,233
410  FCB 0,158,131,0
420  FCB 141,220,0,125
430  FCB 78,0,118,147
440  FCB 0,125,255,0
450  FCB 141,110,0,168
460  FCB 131,0,212,49
470  FCB 0,189,110,0
480  FCB 168,233,0,158
490  FCB 98,0,189,147
500  FCB 0,189,44,0
510  FCB 212,98,0,189
520  FCB 220,0,168,92
530  FCB 0,200,220,0,252
```

O programa tem duas entradas: uma fica em 30000 e outra em 30008 (linha 50). Se a rotina for chamada pelo primeiro endereço, executará a melodia inteira; se for chamada pelo segundo, executará apenas uma nota. Esta última opção permite que se toque a melodia sem interromper a ação do jogo. Podemos tocar uma nota, fazer alguma outra coisa na tela, tocar outra nota, e assim por diante.

ARMAZENE A MELODIA

O valor do rótulo **MUSIC** é colocado no byte rotulado **MUPOINT**. Embora na listagem — e, posteriormente, na memória — o valor de **MUSIC** já esteja nesse byte, usa-se **MUPOINT** para guardar a posição da última nota executada, o que é útil quando tocamos a melodia uma nota de cada vez.

O comando **LDA #19** coloca no acumulador o número de notas do tre-

cho de *Greensleeves* que vamos executar — dezenove. Se você não quiser tocar a melodia inteira de uma vez, coloque no acumulador o número de notas que pretende ouvir. Assim, optando pela execução da música nota por nota, você deverá colocar 1 em A.

O conteúdo de A é levado para a pilha da máquina por **PSHS A**. Trata-se, porém, apenas de uma armazenagem temporária. Mais tarde, o programa precisará do número de notas.

As nove instruções que se seguem viabilizam a produção de sons, colocando os valores apropriados no chip de som do TRS-Color.

A ROTINA PRINCIPAL

MUPOINT aponta para a próxima nota a ser executada, de modo que o endereço do próximo byte da tabela de notas é colocado no apontador da pilha do usuário pelo comando **LDU MUPOINT**. Isto efetivamente transforma a tabela de notas na pilha do usuário.

A instrução **ORCC #$50** impede — "desabilita", no jargão da informática — interrupções, mascarando o conteúdo do registro indicador de status. Quaisquer que sejam os valores dos bits 4 e 6 desse registro, eles passam a ser 1 quando fazemos a operação "ou" entre seu conteúdo e 50 em hexadecimal (80 em decimal).

Como U está apontando para a tabela de notas, a instrução **PULU A,X** traz para dentro de A o próximo byte da tabela, e para dentro de X os dois bytes seguintes. Além disso, também soma 3 ao registro U, que passa a apontar para o próximo byte.

Os dados necessários para o som vêm em grupos de três bytes. O primeiro corresponde ao número de vezes que a nota será tocada. Os dois seguintes determinam a duração que influi na freqüência da nota.

Em seguida, o registro U é comparado com **#MUSIC+57**, o endereço final da tabela de notas. O comando **BLO** provoca um desvio se o valor de U for menor, pois, nesse caso, ainda não chegamos ao fim da tabela e a próxima instrução será ignorada. Ela colocaria o endereço inicial da tabela de volta no apontador e a música poderia, então, ser tocada novamente.

De qualquer maneira, o valor do apontador volta para **MUPOINT**, para que o processador saiba qual é a próxima nota. A duração da nota, que se encontra em X, é colocada temporariamente na pilha da máquina.

As instruções **LDB #252** e **STA $FF20**, referentes ao conversor analógico-digital, preparam-se para usar o altofalante da TV. **LEAX -1,X** diminui o valor de X, e **BNE** faz o processador repetir a instrução anterior, até que X seja zero.

LDX ,S coloca o valor da pilha em X, funcionando como um **PULS X**, sem somar nada, no entanto, ao apontador da pilha. **CLR $FF20** desliga o altofalante da TV e a mesma rotina de contagem em X é repetida.

O valor da pilha retorna a X e o número de vezes que a nota vai ser tocada é diminuído em uma unidade — **DECA**. Outra instrução **BNE** faz o processador repetir o laço **MTWO**, até que A seja reduzido a zero.

A instrução **LEAS 2,S** limpa a pilha e **DEC ,S** subtrai 1 do conteúdo do próximo byte da pilha da máquina — número de notas, colocado na pilha no início. **BNE MAIN** faz o processador voltar e executar a próxima nota, caso o número de notas não tenha sido reduzido a zero.

ANDCC #AF permite novamente — ou seja, "reabilita" — a ocorrência de interrupções, "desmascarando" o registro de status. Finalmente, **PULS A,PC** retira dois valores da pilha da máquina, colocando em A o valor da nota — reduzido a zero — e em **PC**, os próximos dois bytes. Eles correspondem ao endereço de retorno da sub-rotina, colocado ali quando a rotina foi chamada. O procedimento, nesse caso, é equivalente a um comando **RTS**.

Para dotarmos nosso videogame de uma trilha sonora, precisaremos utilizar o microprocessador especial que o MSX possui para produzir sons: o PSG — *Programable Sound Generator*. Apresentamos, no artigo publicado à página

Qual a diferença entre os comandos *outi* e *otir* no MSX?

No artigo da página 213, vimos como empregar o comando de ação em bloco **otir**. Aqui, utilizamos um comando muito parecido: **outi**.

Tanto **otir** como **outi** são usados quando se quer transferir dados que estão em uma tabela na memória RAM, através de uma porta de saída. Esses dois comandos agem sobre os registros HL, B e C. HL é usado como apontador, B como contador e C contém o número da porta de saída. Assim, antes da primeira execução do comando, o par HL deve conter o endereço inicial da tabela, e o registro B, o número de elementos da mesma.

Ambos somam uma unidade ao conteúdo de HL e subtraem o mesmo de B. Mas há uma diferença entre eles: o comando **otir** repete esta operação muitas vezes, transferindo vários caracteres até que B seja reduzido a zero; **outi**, por sua vez, só transfere um número a cada execução.

O comando **otir** é, por si mesmo, um laço completo, utilizando B como contador e fazendo sozinho o teste do indicador de zeros. Mostra-se, portanto, mais adequado quando queremos transferir dados a grande velocidade. No artigo anterior, aqui mencionado, ele foi utilizado para transferir um banco de caracteres para a memória de vídeo.

Já a instrução **outi** deve fazer parte de um laço. Ela vai diminuindo o valor de B, mas não é capaz de descobrir por si mesma que este foi reduzido a zero. Para isso, o programador precisa incluir um teste do tipo **jr nz**. O emprego da instrução **outi** é, assim, mais indicado quando precisamos interromper a transferência dos dados a todo instante para a realização de algum tipo de processamento. Em nosso caso, jamais poderíamos transferir valores para o PSG em alta velocidade — obteríamos barulho, e não música.

A transferência foi feita numa velocidade irregular, sendo interrompida por pausas cuja duração é variável. A execução dessas pausas e a constante mudança nos números dos registros que estavam sendo alterados — enviados por **out 160,a** — correspondem à realização de algum tipo de processamento.

168, uma introdução ao funcionamento desse dispositivo.

O PSG possui catorze registros. Os valores ali colocados determinam os diversos parâmetros do som produzido. Podemos controlar a tonalidade, o volume e sua evolução ao longo do tempo (envoltória ou *envelope*). Temos à nossa disposição três canais, que funcionam simultaneamente, emitindo cada qual um som diferente. Todos eles podem tocar notas musicais e, também, produzir ruídos.

As rotinas Assembly deste artigo executam *Greensleeves* em duas vozes — isto é, são usados dois canais para produzir acordes. Para que o computador saiba que notas tocar e com qual duração, um programa BASIC colocará uma tabela de notas na memória.

CONDIÇÕES INICIAIS DO PSG

A rotina listada a seguir prepara o PSG para tocar a música, colocando valores lidos na tabela dentro dos registros apropriados. Use nosso Assembler para montá-la na memória. Não se esqueça de modificar o comando **CLEAR** da linha 5000, para proteger a memória acima de **&HCFFF**. As rotinas em código devem ser montadas primeiro. Em seguida, pode-se apagar o Assembler e executar o programa BASIC que aparece no final do artigo.

```
10    org -12216
20    ld hl,-14500
30    ld b,6
40    init ld c,160
50    outi
60    ld c,161
70    outi
80    jr nz,init
90    ret
100    end
```

Dispomos de duas alternativas para modificar o valor de um registro do PSG em BASIC. Podemos utilizar tanto o comando **SOUND R,V**— que coloca no registro **R** o valor **V**— quanto a instrução **OUT**.

O microprocessador PSG é considerado um periférico da unidade de processamento central, de maneira que existem dois endereços da área de comunicação — **I/O** — que funcionam como portas de entrada e saída. O valor enviado pela porta 160 indica o registro que será alterado, enquanto a porta 161 corresponde ao destino do novo valor do registro. Dessa maneira, o par de instruções **OUT 160,R:OUT 161,V** equivale ao comando **SOUND R,V**.

Como vimos no artigo da página 213, existe, em linguagem de máquina, uma instrução **out** que funciona de modo parecido ao comando BASIC de mesmo nome. Contudo, como vamos transferir valores de uma tabela para o PSG, usaremos um comando de ação em bloco.

Nossa rotina utiliza o comando **outi**, cujas características devemos entender antes de passarmos a examinar o funcionamento do programa. Esse comando envia o valor contido no endereço apontado pelo par HL através da porta que tem seu número no registro C. Em seguida, o conteúdo do par HL aumenta em uma unidade e o conteúdo de B diminui também em uma unidade.

O comando **outi** é útil para transferir dados de uma tabela, um a um, através de uma porta. Antes de sua primeira aparição, devemos colocar em HL o endereço inicial da tabela, em C, o número da porta, e em B, o número de vezes que queremos repetir o processo. Cada vez que **outi** é executado, HL passa a apontar para o endereço seguinte da tabela. B é diminuído e, quando for reduzido a zero, o indicador de zero será estabelecido.

A primeira instrução determina o endereço inicial da rotina, – 12216. Em seguida, os registros HL, B e C recebem os parâmetros da instrução **outi**. O endereço inicial da tabela de notas é colocado em HL por **ld hl, – 14450**. Como são necessários três pares de valores para ativar o PSG, a linha **ld b,6** coloca 6 em B. A linha rotulada com **init** coloca em C o número da porta de saída que determina o registro.

Quando se executa a instrução **outi** pela primeira vez, o valor que marca o início da tabela, apontado por HL, é enviado pela porta 160 — conteúdo de C. Se olharmos o programa BASIC no final do artigo, veremos que foi enviado o número 7. Soma-se uma unidade ao par HL, que aponta, então, para o próximo número da tabela, e subtrai-se uma unidade de B.

A instrução **ld c,161** faz com que o valor enviado por **outi** na linha seguinte se dirija à porta 161. Se consultarmos novamente a listagem BASIC, veremos que foi enviado o número 24. O valor de HL aumenta novamente e o de B diminui. Como B ainda não foi reduzido a zero, o processador retorna ao rótulo **init**.

O laço entre as linhas 40 e 80 repete-se três vezes. A instrução **outi** é executada seis vezes, de forma que são enviados três pares de números. Subtrai-se de B uma unidade seis vezes. Na sexta vez, a condição de **jr nz,init** não é satisfeita, e a instrução **ret** provoca o retorno da sub-rotina.

Cada volta desse laço equivale a um comando **SOUND**. O primeiro valor enviado é o registro, e o segundo, seu novo conteúdo. Mas que papel desempenha cada registro?

A primeira volta do laço coloca 24 no registro 7 do PSG. Esse registro seleciona o modo de utilização dos canais de som. Cada um dos seus seis primeiros bits determina se um canal vai produzir som, ruído ou se vai permanecer em silêncio. Quando valem 1, os três primeiros bits ativam os canais A, B e C, respectivamente, para a produção de ruído. Os três bits seguintes ativam os canais para a produção de sons musicais. Como 24 é 00011000 em binário, ele ativa a produção de notas musicais em A e B.

Na segunda volta do laço, o valor 15 vai para o registro 8 do PSG. Esse registro controla o volume do canal A — 15 é o volume máximo.

Na terceira e última volta, o registro 9, que controla o volume do canal B do PSG, recebe o valor 15.

TOCANDO UM ACORDE

Depois de ativar o PSG, podemos começar a tocar nossa música. *Greensleeves* compõe-se de acordes de duas notas, cada uma emitida por um dos canais ativados. As notas serão obtidas na mesma tabela, que especifica também qual é a duração do acorde formado por cada par delas. A rotina listada a seguir executa um acorde. Monte-a e grave-a separadamente.

```
10    org -12200
20    ld hl,-14494
30    ld b,4
40    ld c,161
50    ld a,0
60    tune out 160,a
70    inc a
80    outi
90    jr nz,tune
100    ret
110    end
```

Os acordes são compostos por notas emitidas simultaneamente pelos canais A e B. Registros do PSG controlam a tonalidade de cada nota. Os registros 0 e 1 controlam a tonalidade do canal A, enquanto 2 e 3, a do canal B. Os registros 0 e 2 correspondem aos bytes menos significativos; 1 e 3, aos mais significativos. Para emitir um acorde precisaremos, então, modificar o conteúdo de quatro registros do PSG, usando novamente a instrução **outi**. Por isso, a rotina começa acertando os valores de HL, B e C, depois de estabelecido o endereço inicial.

A instrução **ld hl, – 14494** coloca em HL o endereço da primeira nota da tabela. Como emitiremos quatro notas, o número quatro é colocado em B. O número da porta 161 — utilizada para enviar novos valores de registros ao PSG — é colocado em C.

Nesta rotina, o registro A conterá o número do registro do PSG a ser alterado; assim. seu valor inicial será 0 — **ld a,0**

O comando **out 160,a** envia o conteúdo de A pela porta 160, indicando ao PSG o registro que será alterado. Em seguida, A aumenta em uma unidade. A instrução **outi** envia, então, através da porta 161, o próximo valor da tabela, apontado por HL. HL aumenta em uma unidade e B diminui o mesmo tanto. Enquanto B não for reduzido a zero, a instrução **jr nz,tune** repetirá o laço entre as linhas 60 e 90.

Na primeira volta do laço, o registro A contém 0, de maneira que o registro O do PSG recebe o valor da tabela apontado por HL. A cada volta, A aumenta em uma unidade e os registros 1, 2 e 3 do PSG recebem os próximos valores da tabela de notas.

Na quarta vez que **outi** está sendo executado, o valor de B reduz-se a zero. A condição de **jr nz, tune** não é mais satisfeita e a instrução **ret** provoca o retorno da sub-rotina.

CONTROLE DA DURAÇÃO DO ACORDE

O som correspondente ao conteúdo dos registros do PSG é emitido continuamente, até que modifiquemos um de seus parâmetros ou desliguemos o PSG. Assim, para controlarmos a duração do som, usamos uma sub-rotina que não faz absolutamente nada por algum tempo, provocando uma pausa. Enquanto durar essa pausa, o acorde será executado; quando ela terminar, poderemos emitir um novo acorde. A duração será determinada por um número da tabela de notas. Eis a rotina:

```
10    org -12183
20    ld b,(hl)
30    ldp ld hl,1000
40    ld de,0
50    ldq dec hl
60    push hl
70    sbc hl,de
80    pop hl
90    jr nz,ldq
100    djnz ldp
110    ret
120    end
```

Quando a rotina de emissão do acorde termina, deixa em HL o endereço do próximo número da tabela de notas. Esse número indica a duração do acorde,

MICRO DICAS

O PROCESSADOR DO TRS-COLOR

Os usuários do TRS-Color costumam sentir muita dificuldade quando fazem suas primeiras experiências com linguagem de máquina. Para eles, aqui vão algumas recomendações.

O processador 6809 é um dos mais poderosos chips de oito bits. Sua versatilidade tem, contudo, um preço elevado, pois o grande número de comandos e formas de endereçamento pode confundir o principiante.

No 6502 e no Z-80, processadores das outras máquinas, é possível fazer muita coisa interessante com um pequeno conhecimento dos comandos de armazenamento e de controle de laços. Basta saber que valores colocar nas posições adequadas. Programas um pouco maiores exigem o uso da pilha para a armazenagem temporária, mas isto não é um grande problema.

O 6502, processador do Apple e do TK-2000, conta com vários tipos de endereçamento. Mas são tão poucas as instruções disponíveis que essa vantagem deixa de ser significativa.

O 6809 é um precursor dos chips de dezesseis bits. Toda a sua estrutura de endereçamento e armazenagem baseia-se em números de dois bytes. Seus registros comportam dezesseis bits. Ao mesmo tempo, ele é um chip de oito bits. Cria-se, assim, uma infinidade de modos de endereçamentos — surgindo termos como *pós-byte* e *endereço efetivo* —, o que confunde muita gente. São tantos os comandos de ação em bloco, executando tarefas diferentes, que é difícil deduzir suas funções pelo nome.

O que mais confunde o principiante, contudo, é a utilização das pilhas. Há duas pilhas, a do usuário e a da máquina, cada uma com seu apontador. E elas não são usadas apenas para guardar valores temporariamente. O uso inteligente de seus apontadores permite economizar várias linhas de programação, mas certas operações com as pilhas são de deixar qualquer um atordoado. E, pior ainda, é impossível fazer programas sem esses recursos avançados.

A saída está no estudo cuidadoso das instruções explicadas no texto. Na falta de um manual do 6908, liste as características de cada instrução. Outra medida obrigatória é a leitura dos textos introdutórios à programação em código, já publicados em INPUT. Neles o usuário do TRS-Color encontrará conceitos fundamentais sobre o seu processador.

e é colocado em B pela instrução **ld b,(hl)**. O registro B determinará o número de vezes que o laço rotulado **ldp** vai ser executado. Nesse laço, 1000 é colocado no par HL, que diminui uma unidade a cada volta do laço interno **ldq**. Guarda-se HL temporariamente na pilha, para gastar tempo.

A instrução **sbc hl,de** subtrai zero — conteúdo de DE — do conteúdo de HL. Faz-se isto para afetar o indicador de zeros, quando HL for reduzido a zero, pois a instrução **pop HL** não modifica esse sinalizador. Quando HL contém zero, a condição de **jr nz,ldq** não é satisfeita e o processador passa ao laço externo, controlado por B.

A cada volta do laço externo, B diminui. Quando ele se torna zero, **ret** provoca o retorno da sub-rotina.

Podemos controlar o *andamento* da música aumentando ou diminuindo o número de execuções do laço interno. Para isto, basta colocar um número diferente de 1000 em HL, na linha 30.

A ROTINA PRINCIPAL

O trecho de *Greensleeves* que vamos executar tem 88 acordes. As rotinas que mostramos até agora ativaram o PSG e tocaram um acorde apenas. Para a execução da melodia toda, precisaremos de um programa principal, que chame ordenadamente as sub-rotinas. Monte a rotina que se segue com seu Assembler, gravando depois os códigos.

```
10    org -12166
20    call -12216
30    ld b,88
40    loop push bc
50    call -12197
60    push hl
70    call -12183
80    pop hl
90    inc hl
100    pop bc
110    djnz loop
120    call -12213
130    ret
140    end
```

A primeira providência do programa principal é chamar a sub-rotina que ativa o PSG, que fica no endereço −12216. Quando o processador retorna, HL aponta para a primeira nota da música — sétimo número da tabela.

A instrução **ld b,88** coloca em B o número de acordes que serão executados. Como a rotina que emite o acorde utilizará o registro B como contador, o conteúdo deste é temporariamente guardado na pilha com **push BC**.

A rotina que emite o acorde é, então, chamada. Note que o endereço de transferência é −12197, e não −12200 (veja listagem). Ignora-se a primeira linha — linha 20 — da rotina do acorde, pois HL já contém o valor −14494, que corresponde ao sétimo número da tabela. Essa linha foi incluída no programa para que a melodia possa ser tocada mais de uma vez, sem precisarmos ativar novamente o PSG.

Quando o processador retorna da sub-rotina, o par de registros HL aponta para o próximo número da tabela. Como HL será usado como contador na sub-rotina de pausa, seu valor também é guardado na pilha com **push hl**.

A instrução **call −12183** chama a sub-rotina que é responsável pela duração do acorde.

Quando a rotina de pausa termina, o conteúdo de HL é recuperado da pilha. Esse valor, porém, está "atrasado", pois aponta para o número da tabela que controla a duração da nota, já usado pela rotina de pausa. Para que HL aponte para a próxima posição da tabela, precisamos somar 1 ao seu conteúdo, usando **inc hl**.

O conteúdo de B — nosso contador de acordes — é recuperado da pilha por **pop bc**. Em seguida, a instrução **djnz loop** subtrai 1 de B e volta para o rótulo **loop** enquanto B não tiver sido reduzido a zero.

O laço entre as linhas 40 e 110 é repetido 88 vezes, uma para cada acorde. As notas são executadas até que sua tonalidade se modifique — sendo, portanto, substituídas por outras notas. Se o programa parasse aqui, o PSG ficaria emitindo o último acorde da canção durante todo o jogo, o que você certamente não deseja.

Para desligar o PSG, a rotina que o ativou no início é chamada novamente, só que com outro endereço de transferência. A instrução **call − 12213** ignora a primeira linha da sub-rotina, que colocava o endereço inicial da tabela em HL. Assim, como HL ainda aponta para o próximo número não usado na tabela, a sub-rotina utiliza os últimos seis números. Se procurarmos na listagem BASIC, veremos que estes são 8, 0, 9, 0, 10 e 0. Os registros 8, 9 e 10, que controlam o volume dos três canais, recebem, então, o valor 0, e o PSG é silenciado.

Como toda rotina que se preze, esta termina com **ret**.

É interessante notar a importância do apontador HL na execução da melodia. No decorrer da execução, ele percorre toda a tabela. Para que isso seja possível, a cada volta do laço entre as linhas 40 e 110, seu valor é atualizado quatro vezes pela instrução **outi** e uma vez **por inchl** — o que corresponde à transferência de cinco valores da tabela. Quatro deles vão para o PSG e um vai para o registro B controlar a duração da nota. Como a rotina de pausa utiliza HL, entre a execução das linhas 60 e 80 o apontador da posição atual na tabela de notas fica guardado na pilha.

A TABELA DE NOTAS

O conjunto de rotinas apresentado executa uma melodia em duas vozes. Na realidade, ele pode tocar qualquer melodia desse tipo. Cabe à tabela de notas determinar os acordes que serão incluídos. Para criá-la, apague o Assembler e use este programa BASIC.

```
10 CLEAR 200,-14501
20 FOR I=0 TO 451
30 READ A:POKE -14500+I,A
40 NEXT
1000 DATA 7,24,8,15,9,15
1010 DATA 253,0,253,0,10
1020 DATA 213,0,253,0,30
1030 DATA 189,0,213,0,10
1040 DATA 169,0,213,0,15
1050 DATA 159,0,213,0,5
1060 DATA 169,0,28,1,10
```

ERRATA
ASSEMBLER DO TRS-COLOR

Para que seu programa Assembler funcione a contento, você precisará fazer algumas modificações:

```
10 PCLEAR 1:CLEAR 3000:CLS:PRIN
T @233,"INICIALIZANDO":B$=CHR$(
13):POKE 146,1

100 DATA COM,115,3,CWAI,108,1,D
AA,25,,ORA,186,1,TST,125,3,LEAS
,66,3,LEAU,67,3,LEAX,64,3,LEAY,
65,3,MUL,61,,EORA,184,1,ORB,250
,1

1550 P1=1478:P0=P1:P2=0

1560 PRINT @448-P2,K;TAB(6)T$(K
2):P9=P0+LEN(T$(K2))

1565 IF LEN(T$(K2))+P0>1503 THE
N P0=P0-32:P2=P2+32:P1=P1-32:GO
TO 1565

1950 IF X$<>"$" OR BD$<"0" OR B
D$>"G" THEN 1980
```

A alteração na linha 10 é necessária devido aos POKE que aumentam a memória disponível. Se você não executar esses comandos antes de ler o Assembler na fita cassete, ele não conseguirá montar o videogame.

As demais modificações ajustam alguns defeitos contidos no programa original.

```
1070 DATA 189,0,28,1,30
1080 DATA 225,0,225,0,10
1090 DATA 28,1,225,0,15
1100 DATA 253,0,225,0,5
1110 DATA 225,0,253,0,10
1120 DATA 213,0,253,0,30
1130 DATA 253,0,253,0,10
1140 DATA 253,0,63,1,15
1150 DATA 12,1,63,1,5
1160 DATA 253,0,63,1,10
1170 DATA 225,0,253,0,17
1180 DATA 225,0,169,0,10
1190 DATA 12,1,225,0,10
1200 DATA 169,0,82,1,20
1210 DATA 253,0,82,1,10
1220 DATA 213,0,253,0,30
1230 DATA 189,0,253,0,10
1240 DATA 169,0,213,0,15
1250 DATA 159,0,213,0,5
1260 DATA 169,0,213,0,10
1270 DATA 189,0,28,1,30
1280 DATA 225,0,28,1,10
1290 DATA 28,1,225,0,15
1300 DATA 253,0,225,0,5
1310 DATA 225,0,225,0,10
1320 DATA 213,0,253,0,30
1330 DATA 225,0,225,0,5
1340 DATA 253,0,213,0,5
1350 DATA 253,0,189,0,5
1360 DATA 12,1,169,0,15
1370 DATA 45,1,169,0,5
1380 DATA 12,1,169,0,10
1390 DATA 253,0,253,0,30
1400 DATA 253,0,126,0,10
1410 DATA 253,0,169,0,10
1420 DATA 253,0,253,0,30
1430 DATA 142,0,213,0,30
1440 DATA 142,0,189,0,5
1450 DATA 142,0,169,0,5
1460 DATA 142,0,169,0,15
1470 DATA 159,0,189,0,5
1480 DATA 169,0,213,0,10
1490 DATA 189,0,28,1,30
1500 DATA 189,0,253,0,5
1510 DATA 225,0,225,0,10
1520 DATA 28,1,225,0,15
1530 DATA 253,0,253,0,5
1540 DATA 225,0,28,1,10
1550 DATA 213,0,253,0,30
1560 DATA 213,0,225,0,5
1570 DATA 253,0,213,0,10
1580 DATA 253,0,213,0,15
1590 DATA 12,1,225,0,5
1600 DATA 253,0,253,0,10
1610 DATA 225,0,82,1,17
1620 DATA 225,0,169,0,10
1630 DATA 12,1,225,0,10
1640 DATA 82,1,82,1,20
1650 DATA 82,1,123,1,10
1660 DATA 142,0,170,1,30
1670 DATA 142,0,123,1,5
1680 DATA 142,0,82,1,10
1690 DATA 142,0,82,1,15
1700 DATA 159,0,123,1,5
1710 DATA 169,0,170,1,10
1720 DATA 189,0,28,1,30
1730 DATA 189,0,253,0,5
1740 DATA 225,0,225,0,10
1750 DATA 142,0,225,0,15
1760 DATA 253,0,253,0,5
1770 DATA 225,0,28,1,10
1780 DATA 213,0,253,0,30
1790 DATA 225,0,225,0,5
1800 DATA 253,0,213,0,5
1810 DATA 253,0,189,0,5
1820 DATA 12,1,169,0,15
1830 DATA 45,1,169,0,5
1840 DATA 12,1,82,1,10
1850 DATA 253,0,253,0,30
1860 DATA 253,0,126,0,10
1870 DATA 253,0,169,0,10
1880 DATA 253,0,253,0,30
1890 DATA 8,0,9,0,10,0
1900 END
```

A linha 10 protege a memória acima de −14501 para ali colocar a tabela. O laço **FOR...NEXT** entre as linhas 20 e 40 usa **READ** para obter os números da tabela e **POKE** para transferi-los para a memória.

Cada linha **DATA** contém os dados necessários para um acorde. Os números são respectivamente: byte menos significativo da tonalidade do canal A, byte mais significativo dessa mesma tonalidade, byte menos significativo da tonalidade do canal B, byte mais significativo dessa mesma tonalidade e duração do acorde.

Não se preocupe com os valores que determinam as tonalidades das notas. Em um próximo artigo, apresentaremos uma tabela para a conversão dos parâmetros da instrução **PLAY** nos valores de registros do PSG.

O JOGO DO OTELO (2)

Apresentamos neste artigo a segunda metade do jogo do Otelo. Depois de digitá-la, completando o programa, desenvolva algumas jogadas inteligentes... e derrote o computador!

Aqui está a última parte do jogo do Otelo. Após digitar as linhas necessárias para completar a movimentação do jogador, a rotina de movimentação do computador e o final da rotina de jogo, você poderá se considerar pronto para testar suas habilidades, desafiando o seu computador.

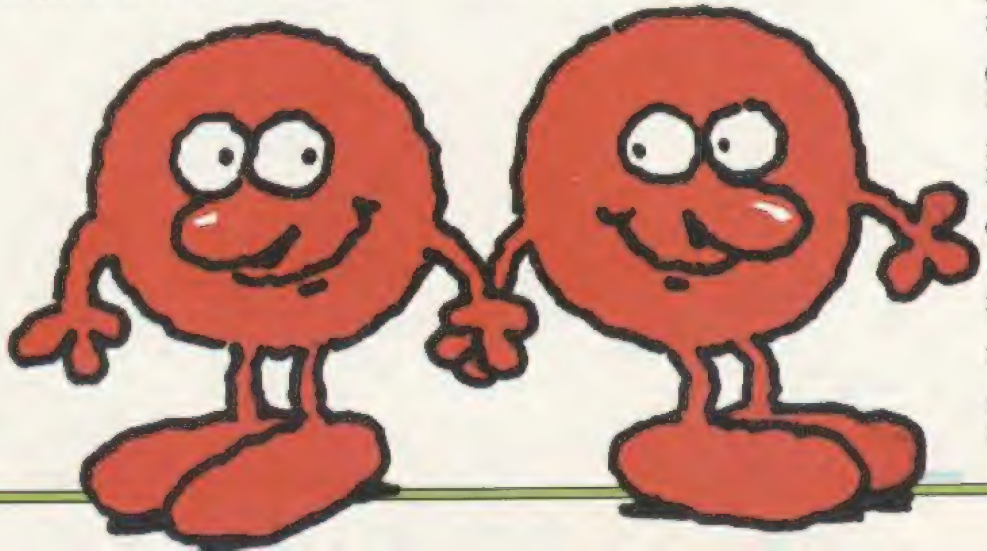
VERIFICAÇÃO DAS POSIÇÕES

```
2090 IF NF=1 THEN 2120
2100 COLOR 1:LINE(0,182)-(255,1
91),PSET,BF:DRAW"C4S8BM0,182":A
$="LONGE DA JOGADA":GOSUB 9300:
FOR F=1 TO 900:NEXT F
2110 GOTO 2000
2120 RF=0:FOR Q=1 TO 8:IF C(Q)=
0 THEN 2170
2130 XP=X:YP=Y
2140 XP=XP+D(Q,1):YP=YP+D(Q,2):
IF XP=0 OR XP=9 OR YP=0 OR YP=9
 THEN C(Q)=0:GOTO 2170
2145 IF B(XP,YP)=2 THEN 2140
2150 IF B(XP,YP)=1 THEN RF=1:GO
TO 2170
2160 IF B(XP,YP)=0 THEN C(Q)=0
2170 NEXT Q
2180 IF RF=1 THEN 2210
2190 COLOR1:LINE(0,182)-(255,19
1),PSET,BF:DRAW"S8C4BM0,182":A$
="JOGADA NAO GANHA A TRILHA":GO
SUB 9300:FOR F=1 TO 900:NEXTF
2200 GOTO 2000
2210 FOR Q=1 TO 8:IF C(Q)=0 THE
N 2250
2220 XP=X+D(Q,1):YP=Y+D(Q,2)
2230 IF B(XP,YP)=1 THEN 2250
2240 B(XP,YP)=1:XP=XP+D(Q,1):YP
=YP+D(Q,2):GOTO 2230
2250 NEXT Q
2260 B(X,Y)=1
2270 CP=2:RETURN
```

```
2090 IF NF=1 THEN  GOTO 2120
2100 PRINT AT 17,0;"LONGE DA JO
GADA": FOR F=1 TO 500: NEXT F
2110 PRINT AT 17,0;"
     ": GOTO 2000
2120 LET RF=0: FOR Q=1 TO 8: IF
 C(Q)=0 THEN  GOTO 2170
2130 LET XP=X: LET YP=Y
2140 LET XP=XP+D(Q,1): LET YP=Y
P+D(Q,2): IF XP=0 OR XP=9 OR YP
=0 OR YP=9 THEN  LET C(Q)=0: GO
TO 2170
2145 IF B(XP,YP)=2 THEN  GOTO 2
140
2150 IF B(XP,YP)=1 THEN  LET RF
=1: GOTO 2170
2160 IF B(XP,YP)=0 THEN  LET C(
Q)=0
2170 NEXT Q
2180 IF RF=1 THEN  GOTO 2210
2190 PRINT AT 17,0;"JOGADA NAO
GANHA A TRILHA": FOR F=1 TO 500
: NEXT F
2200 PRINT AT 17,0;"
                    ": GOTO 2000
2210 FOR Q=1 TO 8: IF C(Q)=0 TH
EN  GOTO 2250
2220 LET XP=X+D(Q,1): LET YP=Y+
D(Q,2)
2230 IF B(XP,YP)=1 THEN  GOTO 2
250
2240 LET B(XP,YP)=1: LET XP=XP+
D(Q,1): LET YP=YP+D(Q,2): GOTO
2230
2250 NEXT Q
2260 LET B(X,Y)=1
2270 LET CP=2: RETURN
```

```
2090 IFNF=1 THEN 2120
2100 LINE(0,182)-(255,191),1,BF
:DRAW"C15S8BM50,182":A$="LONGE
DA JOGADA":GOSUB9300:FORF=1TO90
0:NEXTF
2110 GOTO 2000
2120 RF=0:FORQ=1 TO 8:IFC(Q)=0
THEN2170
2130 XP=X:YP=Y
2140 XP=XP+D(Q,1):YP=YP+D(Q,2):
IFXP=0 OR XP=9 OR YP=0 OR YP=9
THENC(Q)=0:GOTO2170
2145 IF B(XP,YP)=2 THEN 2140
2150 IF B(XP,YP)=1 THEN RF=1:GO
TO 2170
2160 IF B(XP,YP)=0 THEN C(Q)=0
2170 NEXTQ
2180 IF RF=1 THEN 2210
2190 LINE(0,182)-(255,191),1,BF
:DRAW"C15S8BM10,182":A$="JOGADA
 NAO GANHA A TRILHA":GOSUB9300:
FORF=1TO900:NEXTF
2200 GOTO 2000
2210 FORQ=1TO8:IFC(Q)=0THEN2250
2220 XP=X+D(Q,1):YP=Y+D(Q,2)
2230 IF B(XP,YP)=1 THEN2250
2240 B(XP,YP)=1:XP=XP+D(Q,1):YP
=YP+D(Q,2):GOTO2230
2250 NEXTQ
2260 B(X,Y)=1
2270 CP=2:RETURN
```

```
2090  IF NF = 1 THEN 2120
2100  VTAB (17): PRINT "JOGADA
LONGE DAS MINHAS PECAS": FOR F
= 0 TO 1500: NEXT
2110  VTAB (17): PRINT "
                        ": GOTO 2
000
2120 RF = 0: FOR Q = 1 TO 8: IF
 C(Q) = 0 THEN 2170
2130 XP = X:YP = Y
2140 XP = XP + D(Q,1):YP = YP +
 D(Q,2): IF (XP = 0 OR XP = 9)
OR (YP = 0 OR YP = 9) THEN C(Q)
 = 0: GOTO 2170
2145  IF B(XP,YP) = 2 THEN 2140
2150  IF B(XP,YP) = 1 THEN RF =
 1: GOTO 2170
2160  IF B(XP,YP) = 0 THEN C(Q)
 = 0
2170  NEXT
2180  IF RF = 1 THEN 2210
```



- MOVIMENTAÇÃO DO JOGADOR: VERIFICAÇÃO DAS POSIÇÕES
- ROTINA DE MOVIMENTAÇÃO DO COMPUTADOR
- FINAL DA ROTINA DE JOGO
- ANÚNCIO DO VENCEDOR
- OPÇÃO PARA NOVA PARTIDA

```
2190  VTAB (17): PRINT "SUA JOG
ADA NAO GANHA A TRILHA": FOR F
= 0 TO 1500: NEXT
2200  VTAB (17): PRINT "
                      ": GOTO 2
000
2210  FOR Q = 1 TO 8: IF C(Q) =
 0 THEN 2250
2220 XP = X + D(Q,1):YP = Y + D
(Q,2)
2230  IF B(XP,YP) = 1 THEN 2250
2240 B(XP,YP) = 1:XP = XP + D(Q
,1):YP = YP + D(Q,2): GOTO 2230
2250  NEXT Q
2260 B(X,Y) = 1
2270 CP = 2: RETURN
```

Antes de saltar para a linha 2120, o indicador **NF** vê se há alguma peça do computador em um quadrado adjacente (linha 2090). Caso exista, a linha 2100 imprime uma mensagem de erro. As linhas 2120 a 2170 verificam se a jogada encosta a peça em alguma fileira de peças do adversário.

A linha 2140 verifica se a posição checada no passo anterior está dentro dos limites do tabuleiro. Se não estiver, esta posição será abandonada e a próxima, testada. Na linha 2145, testamos se a peça em exame pertence, de fato, ao computador. Em caso afirmativo, o programa volta para a linha 2140 e atualiza a posição.

A linha 2180 examina o resultado da última operação. Se a fileira foi encontrada, o programa salta para a linha 2210. As linhas 2190 e 2200 imprimem uma mensagem no caso de uma fileira não ser ladeada. O programa volta para a linha 2000.

A rotina que descreve a movimentação do jogador está contida entre as linhas 2210 e 2260 do programa. O teste **C(Q) = 0**, da linha 2210, verifica se uma fileira do adversário foi encontrada, por meio de um ciclo que vai de 1 a 8. Se não existir uma fileira naquela direção **(C(Q) = 0)**, o programa salta para a linha 2250 e verifica o **Q** seguinte. A posição do primeiro quadrado da fileira a ser tomada — linha 2040 — é atribuída a **XP** e **YP**.

Na linha 2230, o computador testa se o quadrado em questão é o último da fileira, encerrando-a, portanto. Em caso afirmativo, o programa salta para a linha 2250. Na linha 2240, o programa atribui "um" ao quadrado e, em seguida, volta para a linha 2230, onde testa o próximo quadrado.

Na linha 2260, o quadrado do jogador recebe o valor inicial "um". O indicador **CP** recebe o valor "dois" para o computador e o programa retorna (por meio da linha 2270).

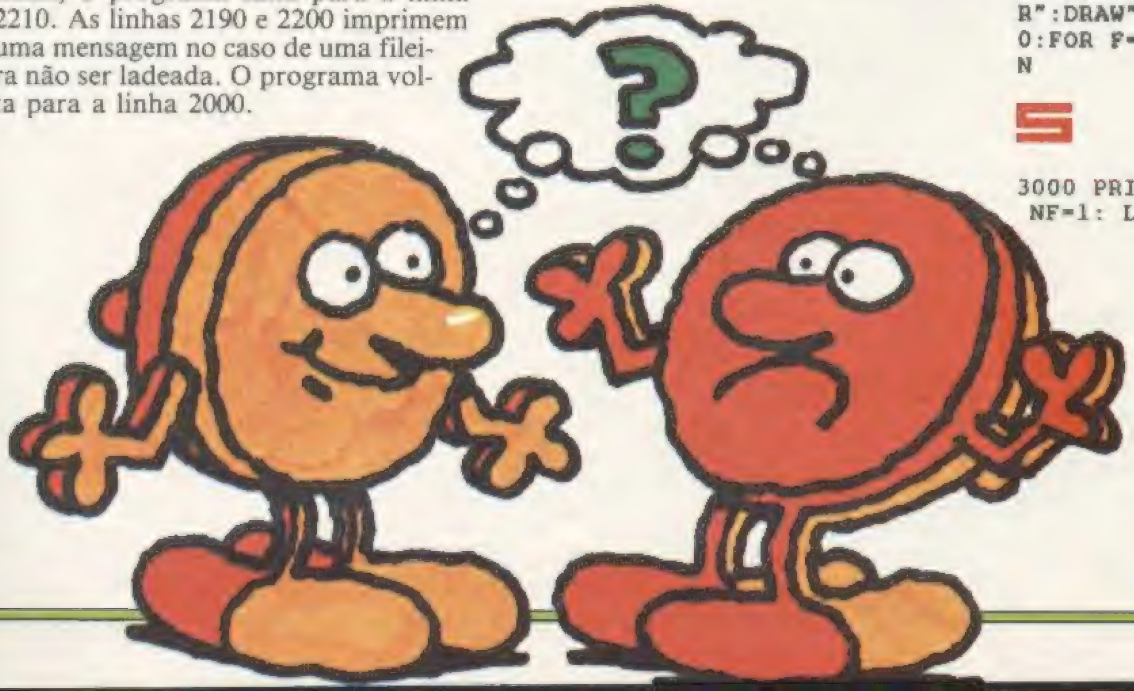
A JOGADA DO COMPUTADOR

T

```
3000 COLOR 1:LINE(0,182)-(255,1
91),PSET,BF:A$="MOVIMENTO DO CO
MPUTADOR":DRAW"S8C3BM26,182":GO
SUB 9300:NF=1:MX=0:FOR X=1 TO 8
:FOR Y=1 TO 8
3010 IF B(X,Y)<>0 THEN 3070
3020 FOR F=1 TO 8:XP=X:YP=Y:DX=
D(F,1):DY=D(F,2):RF=0
3030 XP=XP+DY:YP=YP+DX:IF XP=0
OR XP=9 OR YP=9 THEN 3060
3040 IF B(XP,YP)=1 THEN RF=1:GO
TO 3030
3050 IF B(XP,YP)=2 AND RF=1 THE
N N(NF)=F:X(NF)=X:Y(NF)=Y:NF=NF
+1:F=9
3060 NEXT F
3070 NEXT Y,X:NF=NF-1
3075 IF NF=0 THEN 3300
3080 FOR F=1 TO NF:X=X(F):Y=Y(F
):DX=D(N(F),1):DY=D(N(F),2):CF=
0
3090 X=X+DY:Y=Y+DX:IF B(X,Y)=1
THEN CF=CF+1:GOTO 3090
3100 IF CF>MX THEN MX=CF:MF=F
3110 NEXT F
3180 FOR F=1 TO 8:X=X(MF):Y=Y(M
F):DX=D(F,1):DY=D(F,2)
3190 X=X+DY:Y=Y+DX
3195 IF X<1 OR X>8 OR Y<1 OR Y>
8 THEN 3260
3200 IF B(X,Y)=1 THEN 3190
3210 IF B(X,Y)=2 THEN 3230
3220 IF B(X,Y)=0 THEN 3260
3230 X=X(MF):Y=Y(MF)
3235 B(X,Y)=2:X=X+DY:Y=Y+DX
3240 IF B(X,Y)=2 THEN 3260
3250 GOTO 3235
3260 NEXT F
3265 DRAW"S16;C2;BM118,150"+NU$
(X(MF))+NU$(Y(MF))+"S8"
3270 CP=1:RETURN
3300 COLOR 1:LINE (0,182)-(255,
191),PSET,BF:A$="NAO POSSO MOVE
R":DRAW"S8C4BM50,182":GOSUB 930
0:FOR F=1 TO 50:NEXT:CP=1:RETUR
N
```

S

```
3000 PRINT ''"PENSANDO...": LET
 NF=1: LET MX=0: FOR X=1 TO 8:
```



```
FOR Y=1 TO 8
3010 IF B(X,Y)<>0 THEN  GOTO 30
70
3020 FOR F=1 TO 8: LET XP=X: LE
T YP=Y: LET DX=D(F,1): LET DY=D
(F,2): LET RF=0
3030 LET XP=XP+DY: LET YP=YP+DX
: IF XP=0 OR XP=9 OR YP=0 OR YP
=9 THEN  GOTO 3060
3040 IF B(XP,YP)=1 THEN  LET RF
=1: GOTO 3030
3050 IF B(XP,YP)=2 AND RF=1 THE
N  LET N(NF)=F: LET X(NF)=X: LE
T Y(NF)=Y: LET NF=NF+1: LET F=9
3060 NEXT F
3070 NEXT Y: NEXT X: LET NF=NF-
1
3075 IF NF=0 THEN  GOTO 3300
3080 FOR F=1 TO NF: LET X=X(F):
 LET Y=Y(F): LET DX=D(N(F),1):
LET DY=D(N(F),2): LET CF=0
3090 LET X=X+DY: LET Y=Y+DX: IF
 B(X,Y)=1 THEN  LET CF=CF+1: GO
TO 3090
```

MICRO DICAS

PROGRAMAÇÃO DE MATRIZES

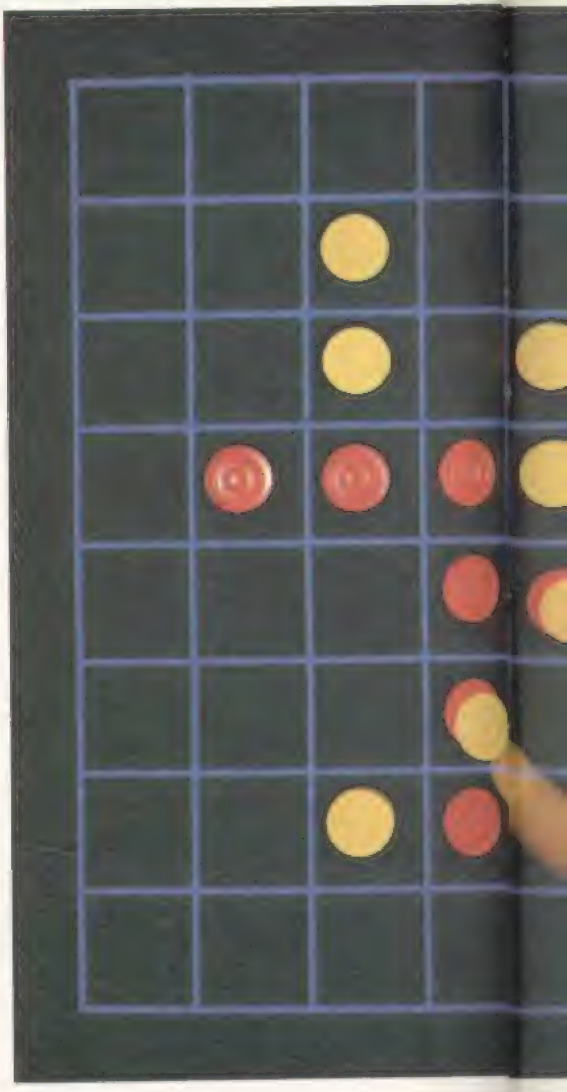
A programação de jogos de tabuleiro em BASIC utiliza quase sempre a mesma técnica: a da programação de matrizes. Uma matriz retangular, dividida em linhas e colunas, é a estrutura de dados ideal para a programação e representação de tabuleiros de xadrez, damas, Otelo, Reversi, Trilha, jogo da velha e outros. Entretanto, jogos que não seguem o esquema de "quadradinhos" identificados em linhas e colunas — como gamão, gomoku e ludo — também podem ser programados através de matrizes.

Em um jogo de tabuleiro retangular, geralmente uma só matriz bidimensional não basta para armazenar todos os dados exigidos. Tomemos como exemplo o xadrez. A cor dos quadrados do tabuleiro pode ser computada por meio de uma equação simples. É mais fácil, porém, armazenar esse tipo de dado com códigos numéricos, em uma matriz COR (8,8). Precisaremos, ainda, de uma matriz para indicar as peças que ocupam cada quadrado (novamente, podemos utilizar códigos, como pião = 1, torre = 2 e assim por diante). Outras matrizes serão necessárias para dados referentes a movimentos, capturas etc.

Tudo isso torna a programação de jogos de tabuleiro mais complexa do que parece. Portanto, se você pretende testar sua habilidade como programador, escolha um jogo bem fácil para começar!

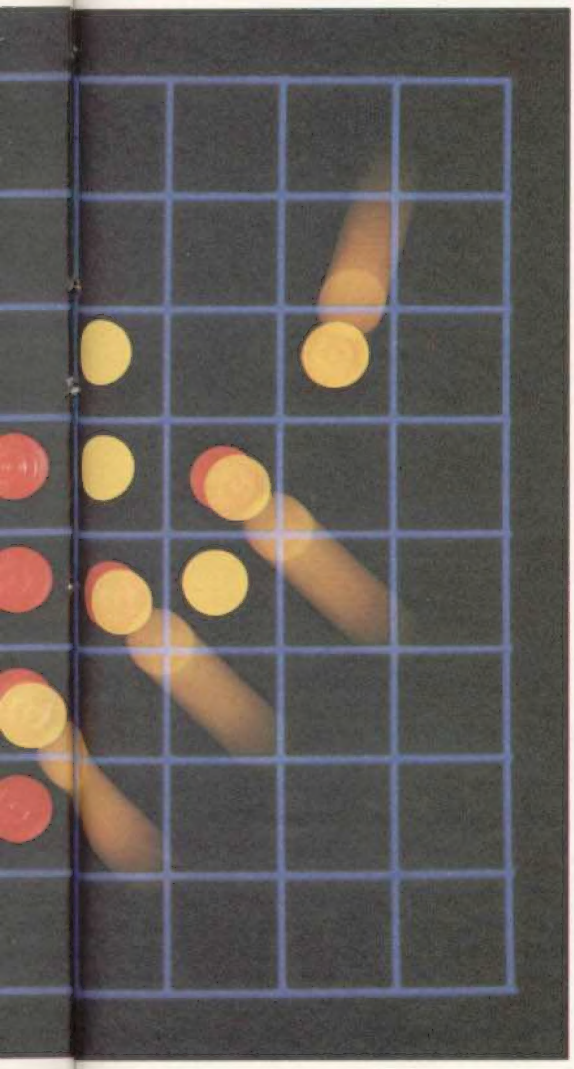
```
3100 IF CF>MX THEN  LET MX=CF:
LET MF=F
3110 NEXT F
3180 FOR F=1 TO 8: LET X=X(MF):
 LET Y=Y(MF): LET DX=D(F,1): LE
T DY=D(F,2)
3190 LET X=X+DY: LET Y=Y+DX
3195 IF X<1 OR X>8 OR Y<1 OR Y>
8 THEN  GOTO 3260
3200 IF B(X,Y)=1 THEN  GOTO 319
0
3210 IF B(X,Y)=2 THEN  GOTO 323
0
3220 IF B(X,Y)=0 THEN  GOTO 326
0
3230 LET X=X(MF): LET Y=Y(MF)
3235 LET B(X,Y)=2: LET X=X+DY:
LET Y=Y+DX
3240 IF B(X,Y)=2 THEN  GOTO 326
0
3250 GOTO 3235
3260 NEXT F
3265 PRINT X(MF),Y(MF): INPUT A
$
3270 LET CP=1: RETURN
3300 PRINT AT 17,0;"NAO POSSO J
OGAR": FOR F=1 TO 500: NEXT F
3305 PRINT AT 17,0;"
                    "
3310 LET CP=1
3320 RETURN
```

```
3000 LINE(0,182)-(255,191),1,BF
:DRAW"S8C15BM30,182":A$="COMPUT
ADOR JOGANDO":GOSUB9300:NF=1:MX
=0:FORX=1TO8:FORY=1TO8
3010 IFB(X,Y)<>0THEN3070
3020 FORF=1TO8:XP=X:YP=Y:DX=D(F
,1):DY=D(F,2):RF=0
3030 XP=XP+DY:YP=YP+DX:IFXP=0 O
R XP=9 OR YP=0 OR YP=9 THEN3060
3040 IFB(XP,YP)=1THENRF=1:GOTO3
030
3050 IFB(XP,YP)=2 AND RF=1 THEN
 N(NF)=F:X(NF)=X:Y(NF)=Y:NF=NF+
1:F=9
3060 NEXTF
3070 NEXTY,X:NF=NF-1
3075 IF NF=0 THEN 3300
3080 FORF=1TONF:X=X(F):Y=Y(F):D
X=D(N(F),1):DY=D(N(F),2):CF=0
3090 X=X+DY:Y=Y+DX:IFB(X,Y)=1TH
ENCF=CF+1:GOTO3090
3100 IFCF>MX THEN MX=CF:MF=F
3110 NEXTF
3180 FORF=1TO8:X=X(MF):Y=Y(MF):
DX=D(F,1):DY=D(F,2)
3190 X=X+DY:Y=Y+DX
3195 IFX<1 OR X>8 OR Y<1 OR Y>8
 THEN3260
3200 IF B(X,Y)=1 THEN 3190
3210 IF B(X,Y)=2 THEN 3230
3220 IF B(X,Y)=0 THEN 3260
3230 X=X(MF):Y=Y(MF)
3235 B(X,Y)=2:X=X+DY:Y=Y+DX
3240 IF B(X,Y)=2 THEN 3260
3250 GOTO 3235
3260 NEXTF
3265 DRAW"S16BM114,154"+NU$(X(M
```



```
F))+NU$(Y(MF))+"S8"
3270 CP=1:RETURN
3300 LINE(0,182)-(255,191),1,BF
:DRAW"C15S8BM50,182":A$="NAO PO
SSO JOGAR":GOSUB9300:FORF=1TO90
0:NEXTF:CP=1:RETURN
```

```
3000 NF = 1:MX = 0: FOR X = 1 T
O 8: FOR Y = 1 TO 8
3010  IF B(X,Y) <  > 0 THEN 307
0
3020  FOR F = 1 TO 8:XP = X:YP
= Y:DX = D(F,1):DY = D(F,2):RF
= 0
3030 XP = XP + DY:YP = YP + DX:
 IF (XP = 0 OR XP = 9) OR (YP =
 0 OR YP = 9) THEN 3060
3040  IF B(XP,YP) = 1 THEN RF =
 1: GOTO 3030
3050  IF B(XP,YP) = 2 AND RF =
1 THEN N(NF) = F:X(NF) = X:Y(NF
) = Y:NF = NF + 1:F = 9
3060  NEXT F
3070  NEXT Y: NEXT X:NF = NF -
1
3075  IF NF = 0 THEN 3300
```



```
3080  FOR F = 1 TO NF:X = X(F):
Y = Y(F):DX = D(N(F),1):DY = D(
N(F),2):CF = 0
3090 X = X + DY:Y = Y + DX: IF
B(X,Y) = 1 THEN CF = CF + 1: GO
TO 3090
3100  IF CF > MX THEN MX = CF:M
F = F
3110  NEXT
3180  FOR F = 1 TO 8:X = X(MF):
Y = Y(MF):DX = D(F,1):DY = D(F,
2)
3190 X = X + DY:Y = Y + DX
3195  IF X < 1 OR X > 8 OR Y <
1 OR Y > 8 THEN 3260
3200  IF B(X,Y) = 1 THEN 3190
3210  IF B(X,Y) = 2 THEN 3230
3220  IF B(X,Y) = 0 THEN 3260
3230 X = X(MF):Y = Y(MF)
3235 B(X,Y) = 2:X = X + DY:Y =
Y + DX
3240  IF B(X,Y) = 2 THEN 3260
3250  GOTO 3235
3260  NEXT
3265  VTAB (17): PRINT "MINHA J
OGADA (LINHA,COLUNA) = ";X(MF);
",";Y(MF): FOR F = 0 TO 3000: N
EXT
3270  VTAB (17): PRINT "
":CP = 1: RETURN
3300  VTAB (17): PRINT "NAO POS
SO JOGAR": FOR F = 0 TO 1500: N
EXT
3310  VTAB (17): PRINT "
":CP = 1: RETURN
```

No começo, o número de quadrados em uma fileira é um, e o número máximo de peças, zero (linha 3000). Dois laços — com variáveis de controle **X** e **Y** — são iniciados, com a função de procurar espaços vazios no tabuleiro. Esta é a parte do programa que consome maior tempo. No decorrer do jogo, à medida que o número de quadrados vazios diminui, o tempo de jogada do computador também vai diminuindo.

A linha 3010 verifica se o quadrado está vazio. Se não estiver, o computador salta os próximos comandos (linha 3070). As linhas que vão de 3020 a 3060 verificam se o quadrado está no fim de uma fileira que poderá ser tomada pelo computador. **XP** e **YP** são usados da mesma maneira que anteriormente. **DX** e **DY** representam os conteúdos do vetor direção **D()** e economizam bastante espaço de memória.

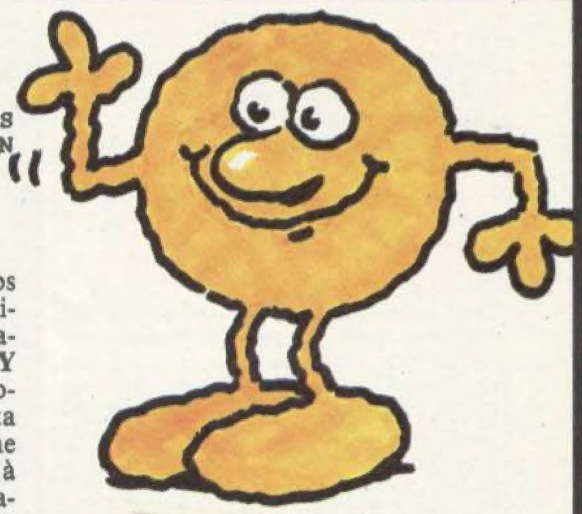
A linha 3030 checa se o quadrado que vai ser testado está dentro dos limites do tabuleiro. Se ele não estiver, a próxima direção será testada. Se o quadrado em exame pertencer ao jogador, a rotina voltará para a linha 3030 para verificar se o quadrado seguinte está ocupado por ele.

A linha 3050 examina o tabuleiro, para saber se está sendo ocupado pelo computador. Caso esteja, e se alguma fileira tiver sido encontrada, as posições iniciais serão gravadas em **X()** e **Y()**, e o número da direção será gravado em **N()**. O contador que indica o número de coordenadas encontradas é também incrementado.

Apenas a primeira fileira será "memorizada", para garantir que a busca leve o menor tempo possível.

As linhas 3080 a 3110 encontram a jogada capaz de fornecer o placar mais longo em uma linha reta. Um ciclo que varia de 1 até **NF** (número de quadrados encontrados) é acionado. **X** e **Y** são equacionados com **X(F)** e **Y(F)**. As coordenadas de direção (**DX** e **DY**) recebem valores iniciais correspondentes às direções indicadas por **N(F)**. Um contador temporário (**CF**) é zerado a cada execução do ciclo.

A linha 3090 verifica se a peça que foi testada é a do jogador. Em caso afirmativo, **CF** é incrementado e o próximo quadrado naquela direção torna-se



objeto de teste. A linha 3100 compara o número encontrado (**CF**) com o valor máximo anterior (**MX**). Se o primeiro for maior, **MX** assume o valor de **CF** e **MF**, as coordenadas da melhor peça encontrada, indicada pelo valor do contador do laço, ou seja, **F**.

As linhas 3180 a 3260 executam a rotina de movimentação. A linha 3180 inicia um ciclo que vai de 1 a 8, de forma que todas as fileiras possam ser encontradas.

As variáveis **X, Y, DX** e **DY** recebem valores iniciais conforme foi explicado anteriormente. A linha 3195 checa se **X** e **Y** ainda estão no tabuleiro. Caso eles não estejam, o programa salta imediatamente para a linha 3260, que contém uma instrução **NEXT**.

A linha 3200 verifica se o jogador está ocupando um quadrado. Se estiver, a rotina salta para tentar o próximo quadrado da fileira. Nenhum quadrado é alterado, pois a rotina está apenas realizando testes.

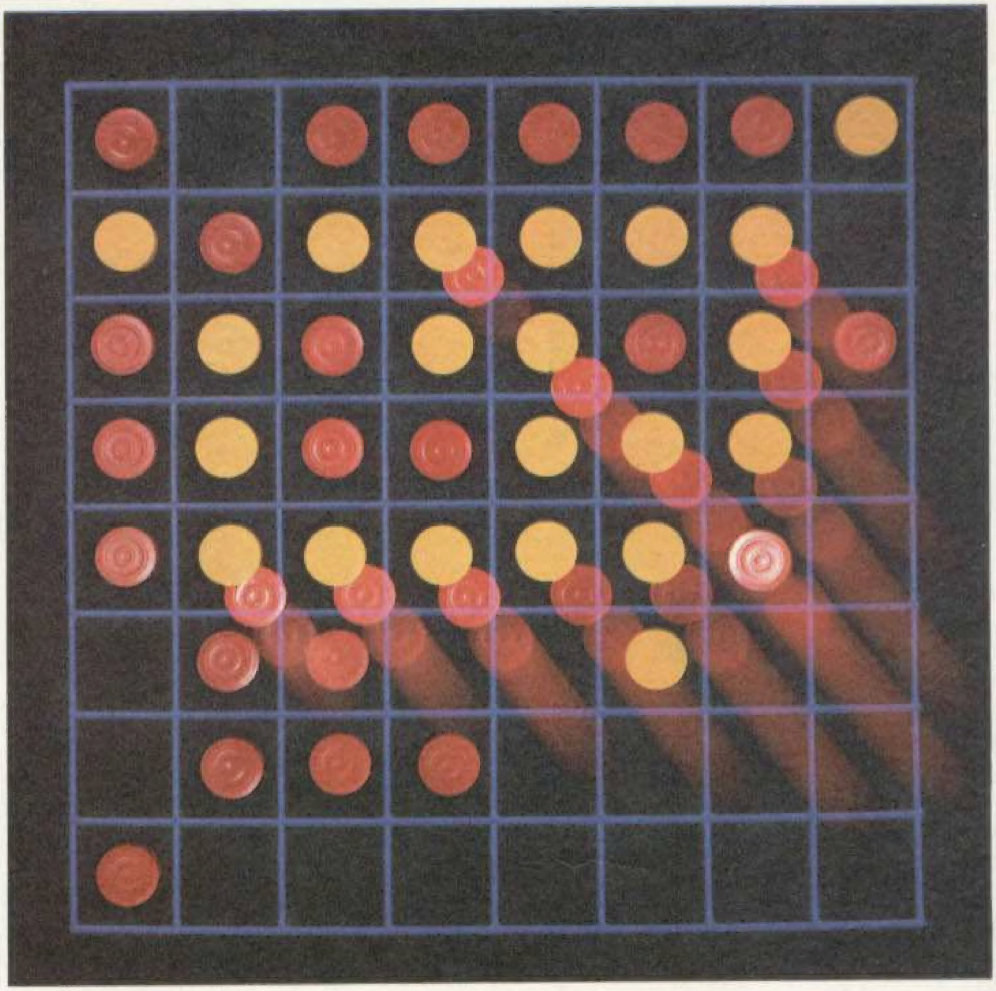
Se a fileira terminar em um quadrado ocupado pelo computador, **X** e **Y** são reinicializadas e as linhas 3235 a 3250 alteram todos os quadrados na fileira. Se um quadrado vazio for encontrado, tenta-se uma nova direção.

Assim que a rotina de movimentação tiver decidido para qual quadrado irá se dirigir, a linha 3265 imprimirá as coordenadas e aguardará que a tecla **<ENTER>** seja pressionada.

Na vez do jogador, o indicador **CP** começa valendo "um" (linha 3270) e, depois, volta para o ciclo principal.

FIM DE JOGO

```
4000 IF PS>CS THEN 5000
4010 IF PS=CS THEN 6000
```



Como adaptar um jogo que utiliza peças coloridas para um micro com vídeo monocromático?

Dependendo do tipo de jogo, a adaptação não é nada fácil. Mas existem algumas alternativas cujo resultado é bastante razoável:

• Para diferenciar as peças dos oponentes, utilize recursos gráficos como hachurados, quadriculados ou pontilhados, cujo efeito é bem visível em preto e branco. Você só terá problemas se as peças forem pequenas a ponto de não haver espaço para nenhum tipo de desenho que as identifique.

• Também como elemento de diferenciação, faça uso do vídeo inverso/vídeo direto — por exemplo, pedras pretas em um quadrado preto, pedras brancas em um quadrado branco e pedras sobre quadrados de outra cor.

• Se o seu computador possui um vídeo monocromático multitonal (isto é, em que as cores têm correspondência em diversas tonalidades de verde ou cinza), identifique as peças dos oponentes utilizando intensidades ou brilhos diferentes.

```
4015 A$="FOI FACIL"
4020 COLOR 1:LINE(0,182)-(255,1
91),PSET,BF:DRAW"S8C2BM70,182":
GOSUB 9300
4025 FOR F=1 TO 1500:NEXT F
4030 COLOR 1:LINE(0,182)-(255,1
91),PSET,BF:A$="QUER JOGAR OUTR
A VEZ ?":DRAW"C3BM0,182":GOSUB
9300
4040 A$=INKEY$:IF A$<>"S" AND A
$<>"N" THEN 4040
4050 IF A$="S" THEN RUN
4060 END
5000 A$="VOCE TEVE SORTE"
5010 GOTO 4020
6000 A$="EMPATAMOS":GOTO 4020
```

```
4000 IF PS>CS THEN  GOTO 5000
4010 IF PS=CS THEN  GOTO 6000
4020 PRINT AT 17,0; INK 2;"FOI
FACIL !"
4030 PRINT "QUER JOGAR OUTRA VE
Z ? (S/N)"
4040 LET A$=INKEY$: IF A$<>"S"
AND A$<>"N" THEN  GOTO 4040
4050 IF A$="S" THEN  RUN
4060 STOP
5000 PRINT AT 17,0; INK 2;"VOCE
 TEVE SORTE !"
5010  GOTO 4030
6000 PRINT AT 17,0; INK 2;"NOS
EMPATAMOS ...                PRE
CISO PRATICAR MAIS !": GOTO 403
0
```

```
4000 IF PS>CS THEN 5000
4010 IF PS=CS THEN 6000
4015 A$="FOI MUITO FACIL"
4020 LINE(0,182)-(255,191),1,BF
:DRAW"C15S8BM70,182":GOSUB9300
4025 FORF=1TO1500:NEXTF
4030 LINE(0,182)-(255,191),1,BF
:A$="QUER JOGAR NOVAMENTE":DRAW
"C15BM30,182":GOSUB9300
4040 A$=INKEY$:IFA$<>"S" AND A$
<>"s" AND A$<>"N" AND A$<>"n" T
HEN 4040
4050 IF A$="S" OR A$="s" THEN R
UN
4060 END
5000 A$="VOCE TEVE SORTE"
5010 GOTO 4020
6000 A$="UM BELO EMPATE":GOTO40
20
```

```
4000  IF PS > CS THEN 5000
4010  IF PS = CS THEN 6000
4020  VTAB (17): PRINT "FOI FAC
IL !"
4030  INPUT "QUER JOGAR NOVAMEN
TE (S/N) ";A$
4040 A$ =  LEFT$ (A$,1): IF A$
< > "S" THEN  END
4050  RUN
4060  END
5000  VTAB (17): PRINT "VOCE TE
VE SORTE DESTA VEZ !"
5010  GOTO 4030
6000  VTAB (17): PRINT "EMPATAM
OS, PRECISO TREINAR MAIS !": GO
TO 4030
```

A rotina de fim de jogo começa na linha 4000, que verifica se o jogador venceu, comparando **PS** com **CS**. O programa salta para a linha 5000 para imprimir a mensagem de vitória. A linha 4010 detecta a ocorrência de empate, e a mensagem correspondente é impressa pela linha 6000. Se o computador tiver vencido, o programa atinge a linha 4020 e exibe uma mensagem para o jogador. As linhas restantes oferecem a opção para uma outra partida.

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR LINHA

LINHA	FABRICANTE	MODELO
Apple II +	Appletronica	Thor 2010
Apple II +	CCE	MC-4000 Exato
Apple II +	CPA	Absolutus
Apple II +	CPA	Polaris
Apple II +	Digitus	DGT-AP
Apple II +	Dismac	D-8100
Apple II +	ENIAC	ENIAC II
Apple II +	Franklin	Franklin
Apple II +	Houston	Houston AP
Apple II +	Magnex	DM II
Apple II +	Maxitronica	MX-2001
Apple II +	Maxitronica	MX-48
Apple II +	Maxitronica	MX-64
Apple II +	Maxitronica	Maxitronic I
Apple II +	Microcraft	Craf II Plus
Apple II +	Milmar	Apple II Plus
Apple II +	Milmar	Apple Master
Apple II +	Milmar	Apple Senior
Apple II +	Omega	MC-400
Apple II +	Polymax	Maxxi
Apple II +	Polymax	Poly Plus
Apple II +	Spectrum	Microengenho I
Apple II +	Spectrum	Spectrum ed
Apple II +	Suporte	Venus II
Apple II +	Sycomig	SIC I
Apple II +	Unitron	AP II
Apple II +	Victor do Brasil	Elppa II Plus
Apple II +	Victor do Brasil	Elppa Jr.
Apple IIe	Microcraft	Craft IIe
Apple IIe	Microdigital	TK-3000 IIe
Apple IIe	Spectrum	Microengenho II
MSX	Gradiente	Expert GPC-1
MSX	Sharp	Hotbit HB-8000
Sinclair Spectrum	Microdigital	TK-90X
Sinclair Spectrum	Timex	Timex 2000
Sinclair ZX-81	Apply	Apply 300
Sinclair ZX-81	Engebras	AS-1000
Sinclair ZX-81	Filcres	NEZ-8000
Sinclair ZX-81	Microdigital	TK-82C
Sinclair ZX-81	Microdigital	TK-83
Sinclair ZX-81	Microdigital	TK-85
Sinclair ZX-81	Prologica	CP-200
Sinclair ZX-81	Ritas	Ringo R-470
Sinclair ZX-81	Timex	Timex 1000
Sinclair ZX-81	Timex	Timex 1500
TRS-80 Mod. I	Dismac	D-8000
TRS-80 Mod. I	Dismac	D-8001/2
TRS-80 Mod. I	LNW	LNW-80
TRS-80 Mod. I	Video Genie	Video Genie I
TRS-80 Mod.III	Digitus	DGT-100
TRS-80 Mod.III	Digitus	DGT-1000
TRS-80 Mod.III	Kemitron	Naja 800
TRS-80 Mod.III	Prologica	CP-300
TRS-80 Mod.III	Prologica	CP-500
TRS-80 Mod.III	Sysdata	Sysdata III
TRS-80 Mod.III	Sysdata	Sysdata Jr.
TRS-80 Mod.IV	Multix	MX-Compacto
TRS-80 Mod.IV	Sysdata	Sysdata IV
TRS-Color	Codimex	CS-6508
TRS-Color	Dynacom	MX-1600
TRS-Color	LZ	Color 64
TRS-Color	Microdigital	TKS-800
TRS-Color	Prologica	CP-400

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR NOME

FABRICANTE	MODELO	PAÍS	LINHA
Appletronica	Thor 2010	Brasil	Apple II +
Apply	Apply 300	Brasil	Sinclair ZX-81
CCE	MC-4000 Exato	Brasil	Apple II +
CPA	Absolutus	Brasil	Apple II +
CPA	Polaris	Brasil	Apple II +
Codimex	CS-6508	Brasil	TRS-Color
Digitus	DGT-100	Brasil	TRS-80 Mod.III
Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Digitus	DGT-AP	Brasil	Apple II +
Dismac	D-8000	Brasil	TRS-80 Mod. I
Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Dismac	D-8100	Brasil	Apple II +
Dynacom	MX-1600	Brasil	TRS-Color
ENIAC	ENIAC II	Brasil	Apple II +
Engebras	AS-1000	Brasil	Sinclair ZX-81
Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Franklin	Franklin	USA	Apple II +
Gradiente	Expert GPC1	Brasil	MSX
Houston	Houston AP	Brasil	Apple II +
Kemitron	Naja 800	Brasil	TRS-80 Mod.III
LNW	LNW-80	USA	TRS-80 Mod. I
LZ	Color 64	Brasil	TRS-Color
Magnex	DM II	Brasil	Apple II +
Maxitronica	MX-2001	Brasil	Apple II +
Maxitronica	MX-48	Brasil	Apple II +
Maxitronica	MX-64	Brasil	Apple II +
Maxitronica	Maxitronic I	Brasil	Apple II +
Microcraft	Craft II Plus	Brasil	Apple II +
Microcraft	Caft IIe	Brasil	Apple IIe
Microdigital	TK-3000 IIe	Brasil	Apple IIe
Microdigital	TK-82C	Brasil	Sinclair ZX-81
Microdigital	TK-83	Brasil	Sinclair ZX-81
Microdigital	TK-85	Brasil	Sinclair ZX-81
Microdigital	TK-90X	Brasil	Sinclair Spectrum
Microdigital	TKS-800	Brasil	TRS-Color
Milmar	Apple II Plus	Brasil	Apple II +
Milmar	Apple Master	Brasil	Apple II +
Milmar	Apple Senior	Brasil	Apple II +
Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Omega	MC-400	Brasil	Apple II +
Polymax	Maxxi	Brasil	Apple II +
Polymax	Poly Plus	Brasil	Apple II +
Prologica	CP-200	Brasil	Sinclair ZX-81
Prologica	CP-300	Brasil	TRS-80 Mod.III
Prologica	CP-400	Brasil	TRS-Color
Prologica	CP-500	Brasil	TRS-80 Mod.III
Ritas	Ringo R-470	Brasil	Sinclair ZX-81
Sharp	Hotbit HB-8000	Brasil	MSX
Spectrum	Microengenho I	Brasil	Apple II +
Spectrum	Microengenho II	Brasil	Apple IIe
Spectrum	Spectrum ed	Brasil	Apple II +
Suporte	Venus II	Brasil	Apple II +
Sycomig	SIC I	Brasil	Apple II +
Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
Timex	Timex 1000	USA	Sinclair ZX-81
Timex	Timex 1500	USA	Sinclair ZX-81
Timex	Timex 2000	USA	Sinclair Spectrum
Unitron	AP II	Brasil	Apple II +
Victor do Brasil	Elppa II Plus	Brasil	Apple II +
Victor do Brasil	Elppa Jr.	Brasil	Apple II +
Video Genie	Video Genie I	USA	TRS-80 Mod. I

UM LOGOTIPO PARA CADA MODELO DE COMPUTADOR

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.
Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:

Sinclair ZX-81

TRS-80

TK-2000

MSX

Spectrum

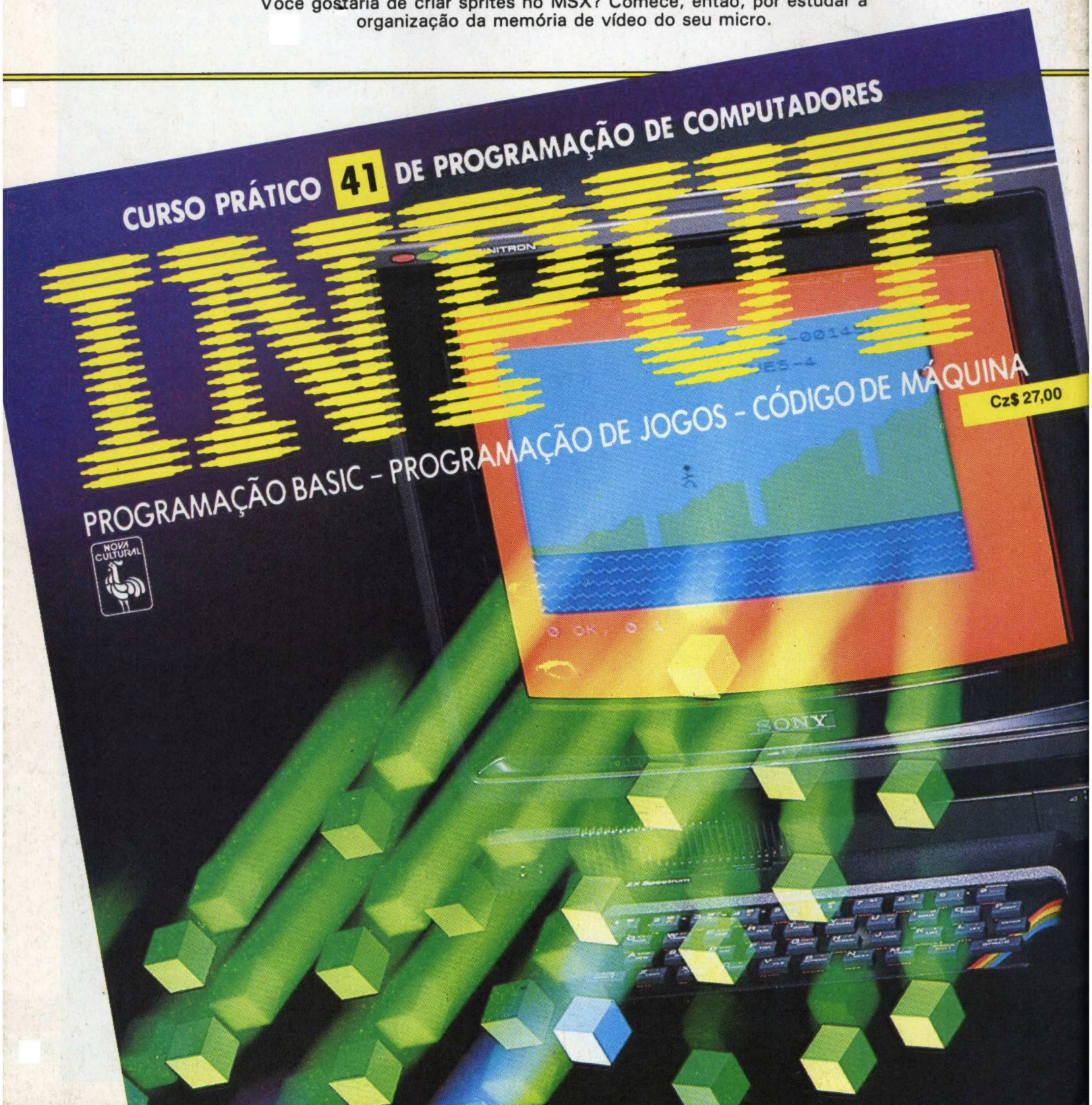
TRS-Color

Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

NO PRÓXIMO NÚMERO

PROGRAMAÇÃO BASIC
As seções do cone. Círculos, elipses, parábolas e hipérboles.
Rotação de curvas. Aplicações práticas.

CÓDIGO DE MÁQUINA
Em busca do lanche perdido, o herói de *Avalanche* deve escalar uma montanha.
Use blocos gráficos para construí-la.

PROGRAMAÇÃO BASIC
Você gostaria de criar sprites no MSX? Comece, então, por estudar a
organização da memória de vídeo do seu micro.